INDUSTRIAL INFORMATION TECHNOLOGY SERIES

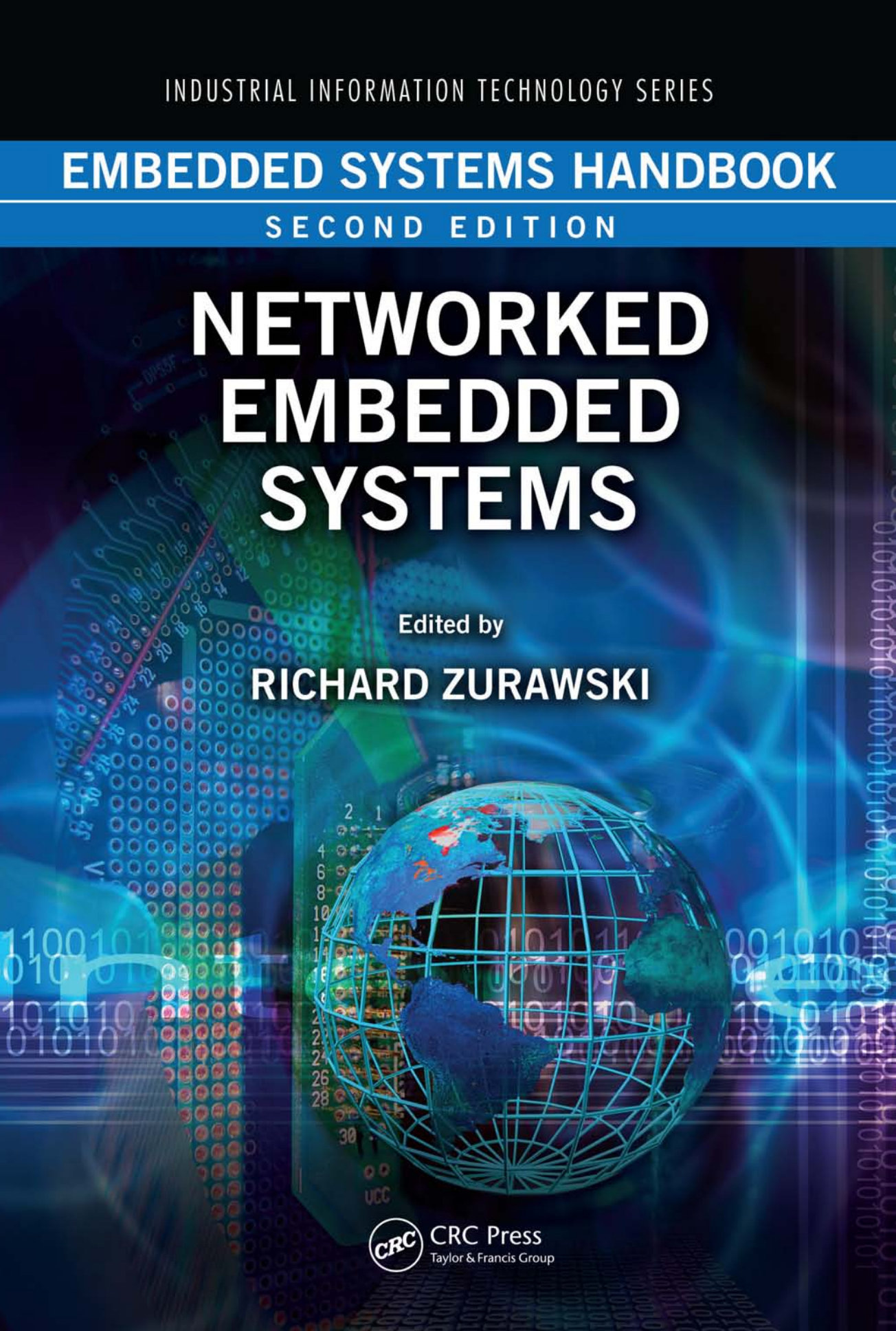
EMBEDDED SYSTEMS HANDBOOK

SECOND EDITION

NETWORKED EMBEDDED SYSTEMS

Edited by

RICHARD ZURAWSKI



CRC Press
Taylor & Francis Group

NETWORKED EMBEDDED SYSTEMS

INDUSTRIAL INFORMATION TECHNOLOGY SERIES

Series Editor
RICHARD ZURAWSKI

Automotive Embedded Systems Handbook
Edited by Nicolas Navet and Françoise Simonot-Lion

Integration Technologies for Industrial Automated Systems
Edited by Richard Zurawski

Electronic Design Automation for Integrated Circuits Handbook
Edited by Luciano Lavagno, Grant Martin, and Lou Scheffer

Embedded Systems Handbook
Edited by Richard Zurawski

Industrial Communication Technology Handbook
Edited by Richard Zurawski

Embedded Systems Handbook, Second Edition
Edited by Richard Zurawski

INDUSTRIAL INFORMATION TECHNOLOGY SERIES

EMBEDDED SYSTEMS HANDBOOK

SECOND EDITION

NETWORKED EMBEDDED SYSTEMS

Edited by

Richard Zurawski

ISA Corporation

San Francisco, California, U.S.A.

CRC Press

Taylor & Francis Group

Boca Raton London New York

CRC Press is an imprint of the

Taylor & Francis Group, an **informa** business

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

© 2009 by Taylor & Francis Group, LLC
CRC Press is an imprint of Taylor & Francis Group, an Informa business

No claim to original U.S. Government works
Printed in the United States of America on acid-free paper
10 9 8 7 6 5 4 3 2 1

International Standard Book Number-13: 978-1-4398-0761-3 (Hardcover)

This book contains information obtained from authentic and highly regarded sources. Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, please access www.copyright.com (http://www.copyright.com/) or contact the Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. CCC is a not-for-profit organization that provides licenses and registration for a variety of users. For organizations that have been granted a photocopy license by the CCC, a separate system of payment has been arranged.

Trademark Notice: Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

Library of Congress Cataloging-in-Publication Data

Embedded systems handbook : embedded systems design and verification / edited by Richard Zurawski. -- 2nd ed.
p. cm. -- (Industrial information technology series ; 6)
Includes bibliographical references and index.
ISBN-13: 978-1-4398-0755-2 (v. 1)
ISBN-10: 1-4398-0755-8 (v. 1)
ISBN-13: 978-1-4398-0761-3 (v. 2)
ISBN-10: 1-4398-0761-2 (v. 2)
1. Embedded computer systems--Handbooks, manuals, etc. I. Zurawski, Richard. II. Title. III. Series.

TK7895.E42E64 2009
004.16--dc22 2008049535

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

Dedication

To Celine, as always.

Contents

Preface xi

Acknowledgments xxvii

Editor xxix

Contributors xxxi

International Advisory Board xxxiii

Part I Network Embedded Systems: An Introduction

1 Networked Embedded Systems: An Overview *Richard Zurawski* **1**-1

2 Middleware Design and Implementation for Networked Embedded Systems *Venkita Subramonian and Christopher D. Gill* **2**-1

Part II Wireless Sensor Networks

3 Introduction to Wireless Sensor Networks *Stefan Dulman and Paul J. M. Havinga* **3**-1

4 Architectures for Wireless Sensor Networks *Stefan Dulman, S. Chatterjea, and Paul J. M. Havinga* **4**-1

5 Overview of Time Synchronization Issues in Sensor Networks *Weilian Su* . **5**-1

6 Resource-Aware Localization in Sensor Networks *Frank Reichenbach, Jan Blumenthal, and Dirk Timmermann* **6**-1

7 Power-Efficient Routing in Wireless Sensor Networks *Lucia Lo Bello and Emanuele Toscano* **7**-1

8 Energy-Efficient MAC Protocols for Wireless Sensor Networks *Lucia Lo Bello, Mario Collotta, and Emanuele Toscano* **8**-1

9 Distributed Signal Processing in Sensor Networks *Omid S. Jahromi and Parham Aarabi* **9**-1

10 Sensor Network Security *Guenter Schaefer* **10**-1

11 Wireless Sensor Networks Testing and Validation *Matthias Woehrle, Jan Beutel, and Lothar Thiele* **11**-1

12 Developing and Testing of Software for Wireless Sensor Networks *Jan Blumenthal, Frank Golatowski, Ralf Behnke, Steffen Prüter, and Dirk Timmermann* . . **12**-1

Part III Automotive Networked Embedded Systems

13 Trends in Automotive Communication Systems *Nicolas Navet and Françoise Simonot-Lion* **13**-1

14 Time-Triggered Communication *Roman Obermaisser* **14**-1

15 Controller Area Networks for Embedded Systems *Gianluca Cena and Adriano Valenzano* **15**-1

16 FlexRay Communication Technology *Roman Nossal-Tueyeni and Dietmar Millinger* **16**-1

17 LIN Standard *Antal Rajnak* **17**-1

18 Standardized System Software for Automotive Applications *Thomas M. Galla* **18**-1

19 Volcano: Enabling Correctness by Design *Antal Rajnak* **19**-1

Part IV Networked Embedded Systems in Industrial Automation

20 Fieldbus Systems: Embedded Networks for Automation *Thilo Sauter* **20**-1

21 Real-Time Ethernet for Automation Applications *Max Felser* **21**-1

22 Configuration and Management of Networked Embedded Devices *Wilfried Elmenreich* **22**-1

23 Networked Control Systems for Manufacturing: Parameterization, Differentiation, Evaluation, and Application *James R. Moyne and Dawn M. Tilbury* . . **23**-1

24 Wireless LAN Technology for the Factory Floor: Challenges and Approaches *Andreas Willig* **24**-1

25 Wireless Local and Wireless Personal Area Network Communication in Industrial Environments *Kirsten Matheus* **25**-1

26 Hybrid Wired/Wireless Real-Time Industrial Networks *Gianluca Cena, Adriano Valenzano, and Stefano Vitturi* **26**-1

27 Wireless Sensor Networks for Automation *Jan-Erik Frey and Tomas Lennvall* **27**-1

28 Design and Implementation of a Truly-Wireless Real-Time Sensor/Actuator Interface for Discrete Manufacturing Automation *Guntram Scheible, Dacfey Dzung, Jan Endresen, and Jan-Erik Frey* **28**-1

Part V Networked Embedded Systems in Building Automation and Control

29 Data Communications for Distributed Building Automation *Wolfgang Kastner and Georg Neugschwandtner* **29**-1

Contributor Index **CI**-1

Subject Index **SI**-1

Preface

Introduction

Application domains have had a considerable impact on the evolution of embedded systems in terms of required methodologies and supporting tools, and resulting technologies. Multimedia and network applications, the most frequently reported implementation case studies at scientific conferences on embedded systems, have had a profound influence on the evolution of embedded systems with the trend now toward multiprocessor systems-on-chip (MPSoCs), which combine the advantages of parallel processing with the high integration levels of systems-on-chip (SoCs). Many SoCs today incorporate tens of interconnected processors; as projected in the 2006 edition of the *International Technology Roadmap for Semiconductors*, the number of processor cores on a chip will reach over 800 by 2020. The design of MPSoCs invariably involves integration of heterogeneous hardware and software IP components, an activity which still lacks a clear theoretical underpinning, and is a focus of many academic and industry projects.

Embedded systems have also been used in automotive electronics, industrial automated systems, building automation and control (BAC), train automation, avionics, and other fields. For instance, trends have emerged for the SoCs to be used in the area of industrial automation to implement complex field-area intelligent devices that integrate the intelligent sensor/actuator functionality by providing on-chip signal conversion, data and signal processing, and communication functions. Similar trends can also be seen in the automotive electronic systems. On the factory floor, microcontrollers are nowadays embedded in field devices such as sensors and actuators. Modern vehicles employ as many as hundreds of microcontrollers. These areas, however, do not receive, for various reasons, as much attention at scientific meetings as the SoC design as it meets demands for computing power posed by digital signal processing (DSP), and network and multimedia processors, for instance.

Most of the mentioned application areas require real-time mode of operation. So do some multimedia devices and gadgets, for clear audio and smooth video. What, then, is the major difference between multimedia and automotive embedded applications, for instance? Braking and steering systems in a vehicle, if implemented as Brake-by-Wire and Steer-by-Wire systems, or a control loop of a high-pressure valve in offshore exploration, are examples of safety-critical systems that require a high level of dependability. These systems must observe hard real-time constraints imposed by the system dynamics, that is, the end-to-end response times must be bounded for safety-critical systems. A violation of this requirement may lead to considerable degradation in the performance of the control system, and other possibly catastrophic consequences. On the other hand, missing audio or video data may result in the user's dissatisfaction with the performance of the system.

Furthermore, in most embedded applications, the nodes tend to be on some sort of a network. There is a clear trend nowadays toward networking embedded nodes. This introduces an additional constraint on the design of this kind of embedded systems: systems comprising a collection of embedded nodes communicating over a network and requiring, in most cases, a high level of dependability. This extra constraint has to do with ensuring that the distributed application tasks execute in a deterministic way (need for application tasks schedulability analysis involving distributed nodes and the communication network), in addition to other requirements such as system availability, reliability, and safety. In general, the design of this kind of networked embedded systems (NES) is a challenge in itself due to the distributed nature of processing elements, sharing common communication medium, and, frequently, safety-critical requirements.

The type of protocol used to interconnect embedded nodes has a decisive impact on whether the system can operate in a deterministic way. For instance, protocols based on random medium access control (MAC) such as carrier sense multiple access (CSMA) are not suitable for this type of operation. On the other hand, time-triggered protocols based on time division multiple access (TDMA) MAC access are particularly well suited for the safety-critical solutions, as they provide deterministic access to the medium. In this category, TTP/C and FlexRay protocols (FlexRay supports a combination of both time-triggered and event-triggered transmissions) are the most notable representatives. Both TTP/C and FlexRay provide additional built-in dependability mechanisms and services which make them particularly suitable for safety-critical systems, such as replicated channels and redundant transmission mechanisms, bus guardians, fault-tolerant clock synchronization, and membership service.

The absence of NES from the academic curriculum is a troubling reality for the industry. The focus is mostly on a single-node design. Specialized networks are seldom mentioned, and if at all, then controller area network (CAN) and FlexRay in the context of embedded automotive systems—a trendy area for examples—but in a superficial way. Specialized communication networks are seldom included in the curriculum of ECE programs. Whatever the reason for this, some engineering graduates involved in the development of embedded systems in diverse application areas will learn the trade the hard way. A similar situation exists with conferences where applications outside multimedia and networking are seldom used as implementation case studies. A notable exception is the IEEE International Symposium on Industrial Embedded Systems that emphasizes research and implementation reports in diverse application areas.

To redress this situation, the second edition of the *Embedded System Handbook* pays considerable attention to the diverse application areas of embedded systems that have in the past few years witnessed an upsurge in research and development, implementation of new technologies, and deployment of actual solutions and technologies. These areas include automotive electronics, industrial automated systems, and BAC. The common denominator for these application areas is their distributed nature and use of specialized communication networks as a fabric for interconnecting embedded nodes.

In automotive electronic systems [1], the electronic control units are networked by means of one of the automotive communication protocols for controlling one of the vehicle functions, for instance, electronic engine control, antilocking brake system, active suspension, and telematics. There are a number of reasons for the automotive industry's interest in adopting field-area networks and mechatronic solutions, known by their generic name as X-by-Wire, aiming to replace mechanical or hydraulic systems by electrical/electronic systems. The main factors seem to be economic in nature, improved reliability of components, and increased functionality to be achieved with a combination of embedded hardware and software. Steer-by-Wire, Brake-by-Wire, or Throttle-by-Wire systems are examples of X-by-Wire systems. The dependability of X-by-Wire systems is one of the main requirements and constraints on the adoption of these kinds of systems. But, it seems that certain safety-critical systems such as Steer-by-Wire and Brake-by-Wire will be complemented with traditional mechanical/hydraulic backups for reasons of safety. Another equally important requirement for X-by-Wire systems is to observe hard real-time constraints imposed by the system dynamics; the end-to-end response times must be bounded for safety-critical systems. A violation of this requirement may lead to degradation in the performance of the control system, and other consequences as a result. Not all automotive electronic systems are safety critical, or require hard real-time response; system(s) to control seats, door locks, internal lights, etc., are some examples. With the automotive industry increasingly keen on adopting mechatronic solutions, it was felt that exploring in detail the design of in-vehicle electronic embedded systems would be of interest to the readers.

In industrial automation, specialized networks [2] connect field devices such as sensors and actuators (with embedded controllers) with field controllers, programmable logic controllers, as well as man–machine interfaces. Ethernet, the backbone technology of office networks, is increasingly being adopted for communication in factories and plants at the field level. The random and native

CSMA/CD arbitration mechanism is being replaced by other solutions allowing for deterministic behavior required in real-time communication to support soft and hard real-time deadlines, time synchronization of activities required to control drives, and for exchange of small data records characteristic of monitoring and control actions. A variety of solutions have been proposed to achieve this goal [3]. The use of wireless links with field devices, such as sensors and actuators, allows for flexible installation and maintenance and mobile operation required in case of mobile robots, and alleviates problems associated with cabling [4]. The area of industrial automation is one of the fastest-growing application areas for embedded systems with thousands of microcontrollers and other electronic components embedded in field devices on the factory floor. This is also one of the most challenging deployment areas for embedded systems due to unique requirements imposed by the industrial environment which considerably differ from those one may be familiar with from multimedia or networking. This application area has received considerable attention in the second edition.

Another fast-growing application area for embedded systems is building automation [5]. Building automation systems aim at the control of the internal environment, as well as the immediate external environment of a building or building complex. At present, the focus of research and technology development is on buildings that are used for commercial purposes such as offices, exhibition centers, and shopping complexes. Some of the main services offered by the building automation systems typically include climate control to include heating, ventilation, and air conditioning; visual comfort to cover artificial lighting; control of daylight; safety services such as fire alarm and emergency sound system; security protection; control of utilities such as power, gas, and water supply; and internal transportation systems such as lifts and escalators.

This books aims at presenting a snapshot of the state-of-the-art embedded systems with an emphasis on their networking and applications. It consists of 48 contributions written by leading experts from industry and academia directly involved in the creation and evolution of the ideas and technologies discussed here. Many of the contributions are from the industry and industrial research establishments at the forefront of developments in embedded systems. The presented material is in the form of tutorials, research surveys, and technology overviews. The contributions are divided into parts for cohesive and comprehensive presentation. The reports on recent technology developments, deployments, and trends frequently cover material released to the profession for the very first time.

Organization

Embedded systems is a vast field encompassing various disciplines. Not every topic, however important, can be covered in a book of a reasonable volume and without superficial treatment. The topics need to be chosen carefully: material for research and reports on novel industrial developments and technologies need to be balanced out; a balance also needs to be struck in treating so-called "core" topics and new trends, and other aspects. The "time-to-market" is another important factor in making these decisions, along with the availability of qualified authors to cover the topics.

This book is divided into two volumes: "Embedded Systems Design and Verification" (Volume I) and "Networked Embedded Systems" (Volume II). Volume I provides a broad introduction to embedded systems design and verification. It covers both fundamental and advanced topics, as well as novel results and approaches, fairly comprehensively. Volume II focuses on NES and selected application areas. It covers the automotive field, industrial automation, and building automation. In addition, it covers wireless sensor networks (WSNs), although from an application-independent viewpoint. The aim of this volume was to introduce actual NES implementations in fast-evolving areas which, for various reasons, have not received proper coverage in other publications. Different application areas, in addition to unique functional requirements, impose specific restrictions on performance, safety, and quality-of-service (QoS) requirements, thus necessitating adoption of different solutions which in turn give rise to a plethora of communication protocols and systems. For this reason, the

discussion of the internode communication aspects has been deferred to this part of the book where the communication aspects are discussed in the context of specific applications of NES.

One of the main objectives of any handbook is to give a well-structured and cohesive description of fundamentals of the area under treatment. It is hoped that Volume I has achieved this objective. Every effort was made to ensure each contribution in this volume contains an introductory material to assist beginners with the navigation through more advanced issues. This volume does not strive to replicate, or replace, university level material. Rather, it tries to address more advanced issues, and recent research and technology developments.

The specifics of the design automation of integrated circuits have been deliberately omitted in this volume to keep it at a reasonable size in view of the publication of another handbook that covers these aspects comprehensively, namely, *The Electronic Design Automation for Integrated Circuits Handbook*, CRC Press, Boca Raton, Florida, 2005, Editors: Lou Scheffer, Luciano Lavagno, and Grant Martin.

The material covered in the second edition of the Embedded Systems Handbook will be of interest to a wide spectrum of professionals and researchers from industry and academia, as well as graduate students from the fields of electrical and computer engineering, computer science and software engineering, and mechatronics engineering.

This edition can be used as a reference (or prescribed text) for university (post) graduate courses. It provides the "core" material on embedded systems. Part II, Volume II, is suitable for a course on WSNs while Parts III and IV, Volume II, can be used for a course on NES with a focus on automotive embedded systems or industrial embedded systems, respectively; this may be complemented with selected material from Volume I.

In the following, the important points of each chapter are presented to assist the reader in identifying material of interest, and to view the topics in a broader context. Where appropriate, a brief explanation of the topic under treatment is provided, particularly for chapters describing novel trends, and for novices in mind.

Volume I. Embedded Systems Design and Verification

Volume I is divided into three parts for quick subject matter identification. Part I, System-Level Design and Verification, provides a broad introduction to embedded systems design and verification covered in 11 chapters: "Real-time in networked embedded systems," "Design of embedded systems," "Models of computation for distributed embedded systems," "Embedded software modeling and design," "Languages for design and verification," "Synchronous hypothesis and polychronous languages," "Processor-centric architecture description languages," "Network-ready, open source operating systems for embedded real-time applications," "Determining bounds on execution times," "Performance analysis of distributed embedded systems," and "Power-aware embedded computing." Part II, Embedded Processors and System-on-Chip Design, gives a comprehensive overview of embedded processors, and various aspects of SoC, FPGA, and design issues. The material is covered in six chapters: "Processors for embedded systems," "System-on-chip design," "SoC communication architectures: From interconnection buses to packet-switched NoCs," "Networks-on-chip: An interconnect fabric for multiprocessor systems-on-chip," "Hardware/software interfaces design for SoC," and "FPGA synthesis and physical design." Part III, Embedded Systems Security and Web Services, gives an overview of "Design issues in secure embedded systems" and "Web services for embedded devices."

Part I. System-Level Design and Verification

An authoritative introduction to real-time systems is provided in the chapter "Real-time in networked embedded systems." This chapter covers extensively the areas of design and analysis with some

examples of analysis and tools; operating systems (an in-depth discussion of real-time embedded operating systems is presented in the chapter "Network-ready, open source operating systems for embedded real-time applications"); scheduling; communications to include descriptions of the ISO/OSI reference model, MAC protocols, networks, and topologies; component-based design; as well as testing and debugging. This is essential reading for anyone interested in the area of real-time systems.

A comprehensive introduction to a design methodology for embedded systems is presented in the chapter "Design of embedded systems." This chapter gives an overview of the design issues and stages. It then presents, in some detail, the functional design; function/architecture and hardware/software codesign; and hardware/software co-verification and hardware simulation. Subsequently, it discusses selected software and hardware implementation issues. While discussing different stages of design and approaches, it also introduces and evaluates supporting tools. This chapter is essential reading for novices for it provides a framework for the discussion of the design issues covered in detail in the subsequent chapters in this part.

Models of computation (MoCs) are essentially abstract representations of computing systems, and facilitate the design and validation stages in the system development. An excellent introduction to the topic of MoCs, particularly for embedded systems, is presented in the chapter "Models of computation for distributed embedded systems." This chapter introduces the origins of MoCs, and their evolution from models of sequential and parallel computation to attempts to model heterogeneous architectures. In the process it discusses, in relative detail, selected nonfunctional properties such as power consumption, component interaction in heterogeneous systems, and time. Subsequently, it reviews different MoCs to include continuous time models, discrete time models, synchronous models, untimed models, data flow process networks, Rendezvous-based models, and heterogeneous MoCs. This chapter also presents a new framework that accommodates MoCs with different timing abstractions, and shows how different time abstractions can serve different purposes and needs. The framework is subsequently used to study coexistence of different computational models, specifically the interfaces between two different MoCs and the refinement of one MoC into another.

Models and tools for embedded software are covered in the chapter "Embedded software modeling and design." This chapter outlines challenges in the development of embedded software, and is followed by an introduction to formal models and languages, and to schedulability analysis. Commercial modeling languages, Unified Modeling Language and Specification and Description Language (SDL), are introduced in quite some detail together with the recent extensions to these two standards. This chapter concludes with an overview of the research work in the area of embedded software design, and methods and tools, such as Ptolemy and Metropolis.

An authoritative introduction to a broad range of design and verification languages used in embedded systems is presented in the chapter "Languages for design and verification." This chapter surveys some of the most representative and widely used languages divided into four main categories: languages for hardware design, for hardware verification, for software, and domain-specific languages. It covers (1) hardware design languages: Verilog, VHDL, and SystemC; (2) hardware verification languages: OpenVera, the e language, Sugar/PSL, and SystemVerilog; (3) software languages: assembly languages for complex instruction set computers, reduced instruction set computers (RISCs), DSPs, and very-long instruction word processors; and for small (4- and 8-bit) microcontrollers, the C and C^{++} Languages, Java, and real-time operating systems; and (4) domain-specific languages: Kahn process networks, synchronous dataflow, Esterel, and SDL. Each group of languages is characterized for their specific application domains, and illustrated with ample code examples.

An in-depth introduction to synchronous languages is presented in the chapter "The synchronous hypothesis and polychronous languages." Before introducing the synchronous languages, this chapter discusses the concept of synchronous hypothesis, the basic notion, mathematical models, and implementation issues. Subsequently, it gives an overview of the structural languages used for modeling and programming synchronous applications, namely, imperative languages Esterel and

SyncCharts that provide constructs to deal with control-dominated programs, and declarative languages Lustre and Signal that are particularly suited for applications based on intensive data computation and dataflow organization. The future trends section discusses loosely synchronized systems, as well as modeling and analysis of polychronous systems and multiclock/polychronous languages.

The chapter "Processor-centric architecture description languages" (ADL) covers state-of-the-art specification languages, tools, and methodologies for processor development used in industry and academia. The discussion of the languages is centered around a classification based on four categories (based on the nature of the information), namely, structural, behavioral, mixed, and partial. Some specific ADLs are overviewed including Machine-Independent Microprogramming Language (MIMOLA); nML; Instruction Set Description Language (ISDL); Machine Description (MDES) and High-Level Machine Description (HMDES); EXPRESSION; and LISA. A substantial part of this chapter focuses on Tensilica Instruction Extension (TIE) ADL and provides a comprehensive introduction to the language illustrating its use with a case study involving design of an audio DSP called the HiFi2 Audio Engine.

An overview of the architectural choices for real-time and networking support adopted by many contemporary operating systems (within the framework of the IEEE 1003.1-2004 international standard) is presented in the chapter "Network-ready, open source operating systems for embedded real-time applications." This chapter gives an overview of several widespread architectural choices for real-time support at the operating system level, and describes the real-time application interface (RTAI) approach in particular. It then summarizes the real-time and networking support specified by the IEEE 1003.1-2004 international standard. Finally, it describes the internal structure of a commonly used open source network protocol stack to show how it can be extended to handle other protocols besides the TCP/IP suite it was originally designed for. The discussion centers on the CAN protocol.

Many embedded systems, particularly hard real-time systems, impose strict restrictions on the execution time of tasks, which are required to complete within certain time bounds. For this class of systems, schedulability analyses require the upper bounds for the execution times of all tasks to be known to verify statically whether the system meets its timing requirements. The chapter "Determining bounds on execution times" presents architecture of the *aiT* timing-analysis tool and an approach to timing analysis implemented in the tool. In the process, it discusses cache-behavior prediction, pipeline analysis, path analysis using integer linear programming, and other issues. The use of this approach is put in the context of upper bounds determination. In addition, this chapter gives a brief overview of other approaches to timing analysis. The validation of nonfunctional requirements of selected implementation aspects such as deadlines, throughputs, buffer space, and power consumption comes under performance analysis.

The chapter "Performance analysis of distributed embedded systems" discusses issues behind performance analysis, and its role in the design process. It also surveys a few selected approaches to performance analysis for distributed embedded systems such as simulation-based methods, holistic scheduling analysis, and compositional methods. Subsequently, this chapter introduces the modular performance analysis approach and accompanying performance networks, as stated by authors, influenced by the worst-case analysis of communication networks. The presented approach allows to obtain upper and lower bounds on quantities such as end-to-end delay and buffer space; it also covers all possible corner cases independent of their probability.

Embedded nodes, or devices, are frequently battery powered. The growing power dissipation, with the increase in density of integrated circuits and clock frequency, has a direct impact on the cost of packaging and cooling, as well as reliability and lifetime. These and other factors make the design for low power consumption a high priority for embedded systems. The chapter "Power-aware embedded computing" presents a survey of design techniques and methodologies aimed at reducing both static and dynamic power dissipation. This chapter discusses energy and power modeling to include instruction-level and function-level power models, microarchitectural power models, memory and bus models, and battery models. Subsequently, it discusses system/application-level optimizations

that explore different task implementations exhibiting different power/energy versus QoS characteristics. Energy-efficient processing subsystems: voltage and frequency scaling, dynamic resource scaling, and processor core selection are addressed next in this chapter. Finally, this chapter discusses energy-efficient memory subsystems: cache hierarchy tuning; novel horizontal and vertical cache partitioning schemes; dynamic scaling of memory elements; software-controlled memories; scratch-pad memories; improving access patterns to on-chip memory; special-purpose memory subsystems for media streaming; and code compression and interconnect optimizations.

Part II. Embedded Processors and System-on-Chip Design

An extensive overview of microprocessors in the context of embedded systems is given in the chapter "Processors for embedded systems." This chapter presents a brief history of embedded microprocessors and covers issues such as software-driven evolution, performance of microprocessors, reduced instruction set computing (RISC) machines, processor cores, and the embedded SoC. After discussing symmetric multiprocessing (SMP) and asymmetric multiprocessing (AMP), this chapter covers some of the most widely used embedded processor architectures followed by a comprehensive presentation of the software development tools for embedded processors. Finally, it overviews benchmarking processors for embedded systems where the use of standard benchmarks and instruction set simulators to evaluate processor cores are discussed. This is particularly relevant to the design of embedded SoC devices where the processor cores may not yet be available in hardware, or be based on user-specified processor configuration and extension.

A comprehensive introduction to the SoC concept, in general, and design issues is provided in the chapter "System-on-chip design." This chapter discusses basics of SoC; IP cores, and virtual components; introduces the concept of architectural platforms and surveys selected industry offerings; provides a comprehensive overview of the SoC design process; and discusses configurable and extensible processors, as well as IP integration quality and certification methods and standards.

On-chip communication architectures are presented in the chapter "SoC communication architectures: From interconnection buses to packet-switched NoCs." This chapter provides an in-depth description and analysis of the three most relevant, from industrial and research viewpoints, architectures to include ARM developed Advanced Micro-Controller Bus Architecture (AMBA) and new interconnect schemes AMBA 3 Advanced eXtensible Interface (AXI), Advanced High-performance Bus (AHB) interface, AMBA 3 APB interface, and AMBA 3 ATB interface; Sonics SMART interconnects (SonicsLX, SonicsMX, and S3220); IBM developed CoreConnect Processor Local Bus (PLB), On-Chip Peripheral Bus (OPB), and Device Control Register (DCR) Bus; and STMicroelectronics developed STBus. In addition, it surveys other architectures such as WishBone, Peripheral Interconnect Bus (PI-Bus), Avalon, and CoreFrame. This chapter also offers some analysis of selected communication architectures. It concludes with a brief discussion of the packet-switched interconnection networks, or Network-on-Chip (NoC), introducing XPipes (a SystemC library of parameterizable, synthesizable NoC components), and giving an overview of the research trends.

Basic principles and guidelines for the NoC design are introduced in the chapter "Networks-on-chip: An interconnect fabric for multiprocessor systems-on-chip." This chapter discusses the rationale for the design paradigm shift of SoC communication architectures from shared busses to NoCs, and briefly surveys related work. Subsequently, it presents details of NoC building blocks to include switch, network interface, and switch-to-switch links. The design principles and the trade-offs are discussed in the context of different implementation variants, supported by the case studies from real-life NoC prototypes. This chapter concludes with a brief overview of NoC design challenges.

The chapter "Hardware/software interfaces design for SoC" presents a component-based design automation approach for MPSoC platforms. It briefly surveys basic concepts of MPSoC design and discusses some related approaches, namely, system-level, platform-based, and component-based.

It provides a comprehensive overview of hardware/software IP integration issues such as bus-based and core-based approaches, integrating software IP, communication synthesis, and IP derivation. The focal point of this chapter is a new component-based design methodology and design environment for the integration of heterogeneous hardware and software IP components. The presented methodology, which adopts automatic communication synthesis approach and uses a high-level API, generates both hardware and software wrappers, as well as a dedicated Operating System for programmable components. The IP integration capabilities of the approach and accompanying software tools are illustrated by redesigning a part of a VDSL modem.

Programmable logic devices, complex programmable logic devices (CPLDs), and field-programmable gate arrays (FPGAs) have evolved from implementing small glue-logic designs to large complete systems that are now the majority of design starts: FPGAs for the higher density design and CPLDs for smaller designs and designs that require nonvolatility targeting. The chapter "FPGA synthesis and physical design" gives an introduction to the architecture of field-programmable date arrays and an overview of the FPGA CAD flow. It then surveys current algorithms for FPGA synthesis, placement, and routing, as well as commercial tools.

Part III. Embedded Systems Security and Web Services

There is a growing trend for networking of embedded systems. Representative examples of such systems can be found in automotive, train, and industrial automation domains. Many of these systems need to be connected to other networks such as LAN, WAN, and the Internet. For instance, there is a growing demand for remote access to process data at the factory floor. This, however, exposes systems to potential security attacks, which may compromise the integrity of the system and cause damage. The limited resources of embedded systems pose considerable challenges for the implementation of effective security policies which, in general, are resource demanding. An excellent introduction to the security issues in embedded systems is presented in the chapter "Design issues in secure embedded systems." This chapter outlines security requirements in computing systems, classifies abilities of attackers, and discusses security implementation levels. Security constraints in embedded systems design discussed include energy considerations, processing power limitations, flexibility and availability requirements, and cost of implementation. Subsequently, this chapter presents the main issues in the design of secure embedded systems. It also covers, in detail, attacks and countermeasures of cryptographic algorithm implementations in embedded systems.

The chapter "Web services for embedded devices" introduces the devices profile for Web services (DPWS). DPWS provides a service-oriented approach for hardware components by enabling Web service capabilities on resource-constraint devices. DPWS addresses announcement and discovery of devices and their services, eventing as a publish/subscribe mechanism, and secure connectivity between devices. This chapter gives a brief introduction to device-centric service-oriented architectures (SOAs), followed by a comprehensive description of DPWS. It also covers software development toolkits and platforms such as the Web services for devices (WS4D), service-oriented architecture for devices (SOA4D), UPnP and DPWS base driver for OSGI, as well as DPWS in Microsoft Vista. The use of DPWS is illustrated by the example of a business-to-business (B2B) maintenance scenario to repair a faulty industrial robot.

Volume II. Networked Embedded Systems

Volume II focuses on selected application areas of NES. It covers automotive field, industrial automation, and building automation. In addition, this volume also covers WSNs, although from an application-independent viewpoint. The aim of this volume was to introduce actual NES

implementations in fast-evolving areas that, for various reasons, have not received proper coverage in other publications. Different application areas, in addition to unique functional requirements, impose specific restrictions on performance, safety, and QoS requirements, thus necessitating adoption of different solutions that in turn give rise to a plethora of communication protocols and systems. For this reason, the discussion of the internode communication aspects has been deferred to this volume where the communication aspects are discussed in the context of specific application domains of NES.

Part I. Networked Embedded Systems: An Introduction

A general overview of NES is presented in the chapter "Networked embedded systems: An overview." It gives an introduction to the concept of NES, their design, internode communication, and other development issues. This chapter also discusses various application areas for NES such as automotive, industrial automation, and building automation.

The topic of middleware for distributed NES is addressed in the chapter "Middleware design and implementation for networked embedded systems." This chapter discusses the role of middleware in NES, and the challenges in design and implementation such as remote communication, location independence, reusing existing infrastructure, providing real-time assurances, providing a robust DOC middleware, reducing middleware footprint, and supporting simulation environments. The focal point of this chapter is the section describing the design and implementation of nORB (a small footprint real-time object request broker tailored to a specific embedded sensor/actuator applications), and the rationale behind the adopted approach.

Part II. Wireless Sensor Networks

The distributed WSN is a relatively new and exciting proposition for collecting sensory data in a variety of environments. The design of this kind of networks poses a particular challenge due to limited computational power and memory size, bandwidth restrictions, power consumption restriction if battery powered (typically the case), communication requirements, and unattended mode of operation in case of inaccessible and/or hostile environments. This part provides a fairly comprehensive discussion of the design issues related to, in particular, self-organizing ad-hoc WSNs. It introduces fundamental concepts behind sensor networks; discusses architectures; time synchronization; energy-efficient distributed localization, routing, and MAC; distributed signal processing; security; testing, and validation; and surveys selected software development approaches, solutions, and tools for large-scale WSNs.

A comprehensive overview of the area of WSNs is provided in the chapter "Introduction to wireless sensor networks." This chapter introduces fundamental concepts, selected application areas, design challenges, and other relevant issues. It also lists companies involved in the development of sensor networks, as well as sensor networks-related research projects.

The chapter "Architectures for wireless sensor networks" provides an excellent introduction to the various aspects of the architecture of WSNs. It starts with a description of a sensor node architecture and its elements: sensor platform, processing unit, communication interface, and power source. It then presents two WSN architectures developed around the layered protocol stack approach, and EYES European project approach. In this context, it introduces a new flexible architecture design approach with environmental dynamics in mind, and aimed at offering maximum flexibility while still adhering to the basic design concept of sensor networks. This chapter concludes with a comprehensive discussion of the distributed data extraction techniques, providing a summary of distributed data extraction techniques for WSNs for the actual projects.

The time synchronization issues in sensor networks are discussed in the chapter "Overview of time synchronization issues in sensor networks." This chapter introduces basics of time synchronization for sensor networks. It also describes design challenges and requirements in developing time synchronization protocols such as the need to be robust and energy aware, the ability to operate correctly in the absence of time servers (server-less), and the need to be lightweight and offer a tunable service. This chapter also overviews factors influencing time synchronization such as temperature, phase noise, frequency noise, asymmetric delays, and clock glitches. Subsequently, different time synchronization protocols are discussed, namely, the network time protocol (NTP), timing-sync protocol for sensor networks (TPSN), H-sensor broadcast synchronization (HBS), time synchronization for high latency (TSHL), reference-broadcast synchronization (RBS), adaptive clock synchronization, time-diffusion synchronization protocol (TDP), rate-based diffusion algorithm, and adaptive-rate synchronization protocol (ARSP).

The localization issues in WSNs are discussed in the chapter "Resource-aware localization in sensor networks." This chapter explains the need to know localization of nodes in a network, introduces distance estimation approaches, and covers positioning and navigation systems as well as localization algorithms. Subsequently, localization algorithms are discussed and evaluated, and are grouped in the following categories: classical methods, proximity based, optimization methods, iterative methods, and pattern matching.

The chapter "Power-efficient routing in wireless sensor networks" provides a comprehensive survey and critical evaluation of energy-efficient routing protocols used in WSNs. This chapter begins by highlighting differences between routing in distributed sensor networks and WSNs. The overview of energy-saving routing protocols for WSNs centers on optimization-based routing protocols, data-centric routing protocols, cluster-based routing protocols, location-based routing protocols, and QoS-enabled routing protocols. In addition, the topology control protocols are discussed.

The chapter "Energy-efficient MAC protocols for wireless sensor networks" provides an overview of energy-efficient MAC protocols for WSNs. This chapter begins with a discussion of selected design issues of the MAC protocols for energy-efficient WSNs. It then gives a comprehensive overview of a number of MAC protocols, including solutions for mobility support and multichannel WSNs. Finally, it outlines current trends and open issues.

Due to their limited resources, sensor nodes frequently provide incomplete information on the objects of their observation. Thus, the complete information has to be reconstructed from data obtained from many nodes frequently providing redundant data. The distributed data fusion is one of the major challenges in sensor networks. The chapter "Distributed signal processing in sensor networks" introduces a novel mathematical model for distributed information fusion which focuses on solving a benchmark signal processing problem (spectrum estimation) using sensor networks.

The chapter "Sensor network security" offers a comprehensive overview of the security issues and solutions. This chapter presents an introduction to selected security challenges in WSNs, such as avoiding and coping with sensor node compromise, maintaining availability of sensor network services, and ensuring confidentiality and integrity of data. Implications of the denial-of-service (DoS) attack, as well as attacks on routing, are then discussed, along with measures and approaches that have been proposed so far against these attacks. Subsequently, it discusses in detail the SNEP and μTESLA protocols for confidentiality and integrity of data, the LEAP protocol, as well as probabilistic key management and its many variants for key management. This chapter concludes with a discussion of secure data aggregation.

The chapter "Wireless sensor networks testing and validation" covers validation and testing methodologies, as well as tools needed to provide support that are essential to arrive at a functionally correct, robust, and long-lasting system at the time of deployment. It explains issues involved in testing of WSNs followed by validation including test platforms and software testing methodologies. An integrated test and instrumentation architecture that augments WSN test beds by incorporating

the environment and giving exact and detailed insight into the reaction to changing parameters and resource usage is then introduced.

The chapter "Developing and testing of software for wireless sensor networks" presents basic concepts related to software development of WSNs, as well as selected software solutions. The solutions include TinyOS, a component-based operating system, and related software packages such as MATÉ, a byte-code interpreter; TinyDB, a query processing system for extracting information from a network of TinyOS sensor nodes; SensorWare, a software framework for WSNs that provides querying, dissemination, and fusion of sensor data, as well as coordination of actuators; Middleware Linking Applications and Networks (MiLAN), a middleware concept that aims to exploit information redundancy provided by sensor nodes; EnviroTrack, a TinyOS-based application that provides a convenient way to program sensor network applications that track activities in their physical environment; SeNeTs, a middleware architecture for WSNs designed to support the pre-deployment phase; Contiki, a lightweight and flexible operating system for 8-bit computers and integrated microcontrollers. This chapter also discusses software solutions for simulation, emulation, and test of large-scale sensor networks: TinyOS SIMulator (TOSSIM), a simulator based on the TinyOS framework; EmStar, a software environment for developing and deploying applications for sensor networks consisting of 32-bit embedded Microserver platforms; SeNeTs, a test and validation environment; and Java-based J-Sim.

Part III. Automotive Networked Embedded Systems

The automotive industry is aggressively adopting mechatronic solutions to replace, or duplicate, existing mechanical/hydraulic systems. The embedded electronic systems together with dedicated communication networks and protocols play a pivotal role in this transition. This part contains seven chapters that offer a comprehensive overview of the area presenting topics such as networks and protocols, operating systems and other middleware, scheduling, safety and fault tolerance, and actual development tools used by the automotive industry.

This part begins with the chapter "Trends in automotive communication systems" that introduces the area of in-vehicle embedded systems and, in particular, the requirements imposed on the communication systems. Then, a comprehensive review of the most widely used, as well as emerging, automotive networks is presented to include priority busses (CAN and J1850), time-triggered networks (TTP/C, TTP/A, TTCAN), low cost automotive networks (LIN and TTP/A), and multimedia networks (MOST and IDB 1394). This is followed by an overview of the industry initiatives related to middleware technologies, with a focus on OSEK/VDX and AUTOSAR.

The chapter "Time-triggered communication" presents an overview of time-triggered communication, solutions, and technologies put in the context of automotive applications. It introduces dependability concepts and fundamental services provided by time-triggered communication protocols, such as clock synchronization, periodic exchange of messages carrying state information, fault isolation mechanisms, and diagnostic services. Subsequently, the chapter overviews four important representatives of time-triggered communication protocols: TTP/C, TTP/A, TTCAN, and TT Ethernet.

A comprehensive introduction to CANs is presented in the chapter "Controller area network." This chapter overviews some of the main features of the CAN protocol, with a focus on advantages and drawbacks affecting application domains, particularly NESs. CANopen, especially suited to NESs, is subsequently covered to include CANopen device profile for generic I/O modules.

The newly emerging standard and technology for automotive safety-critical communication is presented in the chapter "FlexRay communication technology." This chapter overviews aspects such as media access, clock synchronization, startup, coding and physical layer, bus guardian, protocol services, and system configuration.

The Local Interconnect Network (LIN) communication standard, enabling fast and cost-efficient implementation of low-cost multiplex systems for local interconnect networks in vehicles, is presented in the chapter "LIN standard." This chapter introduces the LIN's physical layer and the LIN protocol. It then focuses on the design process and workflow, and covers aspects such as requirement capture (signal definitions and timing requirements), network configuration and design, and network verification, put in the context of Mentor Graphics LIN tool-chain.

The chapter "Standardized basic system software for automotive applications" presents an overview of the automotive software infrastructure standardization efforts and initiatives. This chapter begins with an overview of the automotive hardware architecture. Subsequently, it focuses on the software modules specified by OSEK/VDX and HIS working groups, followed by ISO and AUTOSAR initiatives. Some background and technical information are provided on the Japanese JasPar, the counterpart to AUTOSAR.

The Volcano concept and technology for the design and implementation of in-vehicle networks using the standardized CAN and LIN communication protocols are presented in the chapter "Volcano technology—Enabling correctness by design." This chapter provides an insight in the design and development process of an automotive communication network

Part IV. Networked Embedded Systems in Industrial Automation

Field-Area Networks in Industrial Automation

The advances in design of embedded systems, tools availability, and falling fabrication costs of semiconductor devices and systems allowed for infusion of intelligence into field devices such as sensors and actuators. The controllers used with these devices provide on-chip signal conversion, data and signal processing, and communication functions. The increased functionality, processing, and communication capabilities of controllers have been largely instrumental in the emergence of a widespread trend for networking of field devices around specialized networks, frequently referred to as field-area networks. One of the main reasons for the emergence of field-area networks in the first place was an evolutionary need to replace point-to-point wiring connections by a single bus, thus paving the road to the emergence of distributed systems and, subsequently, NES with the infusion of intelligence into the field devices.

The part begins with a comprehensive introduction to specialized field-area networks presented in the chapter "Fieldbus systems—Embedded networks for automation." This chapter presents evolution of the fieldbus systems; overviews communication fundamentals and introduces the ISO/OSI layered model; covers fieldbus characteristics in comparison with the OSI model; discusses interconnections in the heterogeneous network environment; and introduces industrial Ethernet. Selected fieldbus systems, categorized by the application domain, are summarized at the end. This chapter is a compulsory reading for novices to understand the concepts behind fieldbuses.

The chapter "Real-time Ethernet for automation applications" provides a comprehensive introduction to the standardization process and actual implementation of real-time Ethernet. Standardization process and initiatives, real-time Ethernet requirements, and practical realizations are covered first. The practical realizations discussed include top of TCP/IP, top of Ethernet, and modified Ethernet solutions. Then, this chapter gives an overview of specific solutions in each of those categories.

The issues involved in the configuration (setting up a fieldbus system in the first place) and management (diagnosis and monitoring, and adding new devices to the network) of fieldbus systems are presented in the chapter "Configuration and management of networked embedded devices." This chapter starts by outlining requirements on configuration and management. It then discusses the approach based on the profile concept, as well as several mechanisms following an electronic datasheet approach, namely, the Electronic Device Description Language (EDDL), the Field Device

Tool/Device Type Manager (FDT/DTM), the Transducer Electronic Datasheets (TEDS), and the Smart Transducer Descriptions (STD) of the Interface File System (IFS). It also examines several application development approaches and their influence on the system configuration.

The chapter "Networked control systems for manufacturing: Parameterization, differentiation, evaluation and application" covers extensively the application of networked control systems in manufacturing with an emphasis on control, diagnostics, and safety. It explores the parameterization of networks with respect to balancing QoS capabilities; introduces common network protocol approaches and differentiates them with respect to functional characteristics; presents a method for networked control system evaluation that includes theoretical, experimental, and analytical components; and explores network applications in manufacturing with a focus on control, diagnostics, and safety in general, and at different levels of the factory control hierarchy. Future trends emphasize migration trend toward wireless networking technology.

Wireless Network Technologies in Industrial Automation

Although the use of wireline-based field-area networks is dominant, wireless technology offers a range of incentives in a number of application areas. In industrial automation, for instance, wireless device (sensor/actuator) networks can provide support for mobile operation required for mobile robots, monitoring and control of equipment in hazardous and difficult to access environments, etc. The use of wireless technologies in industrial automation is covered in five chapters that cover the use of wireless local and wireless personal area network technologies on the factory floor, hybrid wired/wireless networks in industrial real-time applications, a wireless sensor/actuator (WISA) network developed by ABB and deployed in a manufacturing environment, and WSNs for automation.

The issues involving the use of wireless technologies and mobile communication in the industrial environment (factory floor) are discussed in the chapter "Wireless LAN technology for the factory floor: Challenges and approaches." This is comprehensive material dealing with topics such as error characteristics of wireless links and lower layer wireless protocols for industrial applications. It also briefly discusses hybrid systems extending selected fieldbus technologies (such as PROFIBUS and CAN) with wireless technologies.

The chapter "Wireless local and wireless personal area network communication in industrial environments" presents a comprehensive overview of the commercial-off-the-shelf wireless technologies to include IEEE 802.15.1/Bluetooth, IEEE 802.15.4/ZigBee, and IEEE 802.11 variants. The suitability of these technologies for industrial deployment is evaluated to include aspects such as application scenarios and environments, coexistence of wireless technologies, and implementation of wireless fieldbus services.

Hybrid configurations of communication networks resulting from wireless extensions of conventional, wired, industrial networks and their evaluation are presented in the chapter "Hybrid wired/wireless real-time industrial networks." The focus is on four popular solutions, namely, Profibus DP and DeviceNet, and two real-time Ethernet networks: Profinet IO and EtherNet/IP; and the IEEE 802.11 family of WLAN standards and IEEE 802.15.4 WSNs as wireless extensions. They are some of the most promising technologies for use in industrial automation and control applications, and a lot of devices are already available off-the-shelf at relatively low cost.

The chapter "Wireless sensor networks for automation" gives a comprehensive introduction to WSNs technology in embedded applications on the factory floor and other industrial automated systems. This chapter gives an overview of WSNs in industrial applications; development challenges; communication standards including ZeegBee, WirelessHART, and ISA100; low-power design; packaging of sensors and ICs; software/hardware modularity in design, and power supplies. This is essential reading for anyone interested in wireless sensor technology in factory and industrial automated applications.

A comprehensive case study of a factory-floor deployed WSN is presented in the chapter "Design and implementation of a truly wireless real-time sensor/actuator interface for discrete manufacturing automation." The system, known as WISA has been implemented by ABB in a manufacturing cell to network proximity switches. The sensor/actuators communication hardware is based on a standard Bluetooth 2.4 GHz radio transceiver and low-power electronics that handle the wireless communication link. The sensors communicate with a wireless base station via antennas mounted in the cell. For the base station, a specialized RF front end was developed to provide collision-free air access by allocating a fixed TDMA time slot to each sensor/actuator. Frequency hopping (FH) was employed to counter both frequency-selective fading and interference effects, and operates in combination with automatic retransmission requests (ARQ). The parameters of this TDMA/FH scheme were chosen to satisfy the requirements of up to 120 sensor/actuators per base station. Each wireless node has a response or cycle time of 2 ms, to make full use of the available radio band of 80 MHz width. The FH sequences are cell-specific and were chosen to have low cross-correlations to permit parallel operation of many cells on the same factory floor with low self-interference. The base station can handle up to 120 WISAs and is connected to the control system via a (wireline) field bus. To increase capacity, a number of base stations can operate in the same area. WISA provides wireless power supply to the sensors, based on magnetic coupling.

Part V. Networked Embedded Systems in Building Automation and Control

Another fast-growing application area for NES is BAC. BAC systems aim at the control of the internal environment, as well as the immediate external environment of a building or building complex. At present, the focus of research and technology development is on buildings that are used for commercial purposes such as offices, exhibition centers, and shopping complexes. However, the interest in (family type) home automation is on the rise.

A general overview of the building control and automation area and the supporting communication infrastructure is presented in the chapter "Data communications for distributed building automation." This chapter provides an extensive description of building service domains and the concepts of BAC, and introduces building automation hierarchy together with the communication infrastructure. The discussion of control networks for building automation covers aspects such as selected QoS requirements and related mechanisms, horizontal and vertical communication, network architecture, and internetworking. As with industrial fieldbus systems, there are a number of bodies involved in the standardization of technologies for building automation. This chapter overviews some of the standardization activities, standards, as well as networking and integration technologies. Open systems BACnet, LonWorks, and EIB/KNX, wireless IEEE 802.15.4 and ZigBee, and Web Services are introduced at the end of this chapter, together with a brief introduction to home automation.

References

1. N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, Trends in automotive communication systems, *Special Issue: Industrial Communication Systems*, R. Zurawski, Ed., *Proceedings of the IEEE*, 93(6), June 2005, 1204–1223.
2. J.-P. Thomesse, Fieldbus technology in industrial automation, *Special Issue: Industrial Communication Systems*, R. Zurawski, Ed., *Proceedings of the IEEE*, 93(6), June 2005, 1073–1101.
3. M. Felser, Real-time Ethernet—Industry perspective, *Special Issue: Industrial Communication Systems*, R. Zurawski, Ed., *Proceedings of the IEEE*, 93(6), June 2005, 1118–1129.

4. A. Willig, K. Matheus, and A. Wolisz, Wireless technology in industrial networks, *Special Issue: Industrial Communication Systems*, R. Zurawski, Ed., *Proceedings of the IEEE*, 93(6), June 2005, 1130–1151.
5. W. Kastner, G. Neugschwandtner, S. Soucek, and H. M. Newman, Communication systems for building automation and control, *Special Issue: Industrial Communication Systems*, R. Zurawski, Ed., *Proceedings of the IEEE*, 93(6), June 2005, 1178–1203.

Locating Topics

To assist readers with locating material, a complete table of contents is presented at the front of the book. Each chapter begins with its own table of contents. Two indexes are provided at the end of the book. The index of authors contributing to the book together with the titles of the contributions, and a detailed subject index.

Richard Zurawski

Acknowledgments

I would like to thank all authors for the effort to prepare the contributions and tremendous cooperation. I would like to express gratitude to my publisher Nora Konopka and other CRC Press staff involved in the book production. My love goes to my wife who tolerated the countless hours I spent on preparing this book.

Richard Zurawski

Editor

Richard Zurawski is with ISA Group, San Francisco, California, involved in providing solutions to 1000 Fortune companies. He has over 30 years of academic and industrial experience, including a regular professorial appointment at the Institute of Industrial Sciences, University of Tokyo, and full-time R&D advisor with Kawasaki Electric, Tokyo. He has provided consulting services to Kawasaki Electric, Ricoh, and Toshiba Corporations, Japan. He has participated in a number of Japanese Intelligent Manufacturing Systems programs.

Dr. Zurawski's involvement in R&D and consulting projects and activities in the past few years included network-based solutions for factory floor control, network-based demand side management, Java technology, SEMI implementations, wireless applications, IC design and verification, EDA, and embedded systems integration.

Dr. Zurawski is the series editor for The Industrial Information Technology (book) Series, CRC Press/Taylor & Francis; and the editor in chief of the *IEEE Transactions on Industrial Informatics*. He has served as editor at large for *IEEE Transactions on Industrial Informatics* (2005–2006); and as an associate editor for *IEEE Transactions on Industrial Electronics* (1994–2005); *Real-Time Systems*; *The International Journal of Time-Critical Computing Systems*, Kluwer Academic Publishers (1997–2003); and *The International Journal of Intelligent Control and Systems*, World Scientific Publishing Company (1996–1997).

Dr. Zurawski was a guest editor of three special issues in *IEEE Transactions on Industrial Electronics* on factory automation and factory communication systems. He was also a guest editor of the special issue on industrial communication systems in the *Proceedings of the IEEE*. He was invited by *IEEE Spectrum* to contribute an article on Java technology to "Technology 1999: Analysis and Forecast" special issue.

Dr. Zurawski served as a vice president of the Industrial Electronics Society (IES) (1994–1997), as a chairman of the IES Factory Automation Council (1994–1997), and is currently the chairman of the IES Technical Committee on Factory Automation. He was also on a steering committee of the *ASME/IEEE Journal of Microelectromechanical Systems*. In 1996, he received the Anthony J. Hornfeck Service Award from the IEEE IES.

Dr. Zurawski has served as a general co-chair for 13 IEEE conferences and workshops, as a technical program co-chair for 4 IEEE conferences, as a track (co-)chair for 12 IEEE conferences, and as a member of program committees for over 40 IEEE, IFAC, and other conferences and workshops. He has established two major technical events: IEEE Workshop on Factory Communication Systems and IEEE International Conference on Emerging Technologies and Factory Automation.

Dr. Zurawski was the editor of five major handbooks: *The Industrial Information Technology Handbook*, CRC Press, Boca Raton, Florida, 2004; *The Industrial Communication Technology Handbook*, CRC Press, Boca Raton, Florida, 2005; *The Embedded Systems Handbook*, CRC Press/Taylor & Francis, Boca Raton, Florida, 2005; *Integration Technologies for Industrial Automated Systems*, CRC Press/Taylor & Francis, Boca Raton, Florida, 2006; and *Networked Embedded Systems Handbook*, CRC Press/Taylor & Francis, Boca Raton, Florida, 2008.

Dr. Zurawski received his MEng in electronics from the University of Mining and Metallurgy, Krakow and PhD in computer science from LaTrobe University, Melbourne, Australia.

Contributors

Parham Aarabi
Edward S. Rogers Sr. Department of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario, Canada

Ralf Behnke
Institute of Applied Microelectronics and Computer Engineering
University of Rostock
Rostock, Germany

Jan Beutel
Computer Engineering and Networks Laboratory
Swiss Federal Institute of Technology Zurich
Zurich, Switzerland

Jan Blumenthal
Embedding Innovations
SYSGO AG
Klein-Winternheim, Germany

Gianluca Cena
Institute of Electronics and Information Engineering and Telecommunications
National Research Council
Turin, Italy

S. Chatterjea
Electrical Engineering, Mathematics and Computer Science
University of Twente
Enschede, the Netherlands

Mario Collotta
Department of Computer Engineering and Telecommunications
University of Catania
Catania, Italy

Stefan Dulman
Electrical Engineering, Mathematics and Computer Science
University of Twente
Enschede, the Netherlands

Dacfey Dzung
Corporate Research
ABB Switzerland Limited
Baden-Daettwil, Switzerland

Wilfried Elmenreich
Institute of Networked and Embedded Systems
University of Klagenfurt
Klagenfurt, Austria

Jan Endresen
ABB Corporate Research
Billingstad, Norway

Max Felser
Engineering and Information Technology
Bern University of Applied Sciences
Burgdorf, Switzerland

Jan-Erik Frey
ABB Corporate Research
Västerås, Sweden

Thomas M. Galla
Elektrobit Austria GmbH
Vienna, Austria

Christopher D. Gill
Department of Computer Science and Engineering
Washington University
St. Louis, Missouri

Frank Golatowski
Institute of Applied Microelectronics and Computer Engineering
University of Rostock
Rostock, Germany

Paul J. M. Havinga
Electrical Engineering, Mathematics and Computer Science
University of Twente
Enschede, the Netherlands

Omid S. Jahromi
Sonavation Inc.
Palm Beach Gardens, Florida

Wolfgang Kastner
Institute of Computer Aided Automation
Vienna University of Technology
Vienna, Austria

Tomas Lennvall
ABB Corporate Research
Västerås, Sweden

Lucia Lo Bello
Department of Computer Engineering and Telecommunications
University of Catania
Catania, Italy

Kirsten Matheus
NXP Semiconductors
Hamburg, Germany

Dietmar Millinger
Elektrobit Austria GmbH
Vienna, Austria

James R. Moyne
Department of Mechanical Engineering
University of Michigan
Ann Arbor, Michigan

Nicolas Navet
National Institute for Research in Computer Science and Control (INRIA)—RealTime-at-Work
Vandoeuvre-lès-Nancy, France

Georg Neugschwandtner
Institute of Computer Aided Automation
Vienna University of Technology
Vienna, Austria

Roman Nossal-Tueyeni
Austro Control GmbH
Vienna, Austria

Roman Obermaisser
Institute of Computer Engineering
Vienna University of Technology
Vienna, Austria

Steffen Prüter
Institute of Applied Microelectronics and Computer Engineering
University of Rostock
Rostock, Germany

Antal Rajnak
System Level Engineering Division
Mentor Graphics Corporation
Geneva, Switzerland

Frank Reichenbach
Wireless & Embedded Systems
ABB Corporate Research
Billingstad, Norway

Thilo Sauter
Research Unit for Integrated Sensor Systems
Austrian Academy of Sciences
Wiener Neustadt, Austria

Guenter Schaefer
Telematics/Computer Networks Group
Ilmenau University of Technology
Ilmenau, Germany

Guntram Scheible
ABB Stotz-Kontakt GmbH
Heidelberg, Germany

Françoise Simonot-Lion
Lorraine Laboratory of Computer Science Research and Applications (LORIA)—University of Nancy
Vandoeuvre-lès-Nancy, France

Weilian Su
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, California

Venkita Subramonian
AT&T Labs Research
Florham Park, New Jersey

Lothar Thiele
Computer Engineering and Networks Laboratory
Swiss Federal Institute of Technology Zurich
Zurich, Switzerland

Dawn M. Tilbury
Department of Mechanical Engineering
and
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, Michigan

Dirk Timmermann
Institute of Applied Microelectronics and Computer Engineering
University of Rostock
Rostock, Germany

Emanuele Toscano
Department of Computer Engineering and Telecommunications
University of Catania
Catania, Italy

Adriano Valenzano
Institute of Electronics and Information Engineering and Telecommunications
National Research Council
Turin, Italy

Stefano Vitturi
Institute of Electronics and Information Engineering and Telecommunications
National Research Council
Turin, Italy

Andreas Willig
Telecommunication Networks Group
Technical University of Berlin
Berlin, Germany

Matthias Woehrle
Computer Engineering and Networks Laboratory
Swiss Federal Institute of Technology Zurich
Zurich, Switzerland

Richard Zurawski
ISA Group
Alameda, California

International Advisory Board

Alberto Sangiovanni-Vincentelli, University of California, Berkeley, California (Chair)
Giovanni De Michelli, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
Robert de Simone, National Institute for Research in Computer Science and Control (INRIA), Sophia Antipolis, France
Stephen A. Edwards, Columbia University, New York, New York
Rajesh Gupta, University of California, San Diego, California
Axel Jantsch, Royal Institute of Technology, Stockholm, Sweden
Wido Kruijtzer, Philips Research, Eindhoven, The Netherlands
Luciano Lavagno, Polytechnic University of Turin, Turin, Italy and Cadence Berkeley Labs, Berkeley, California
Grant Martin, Tensilica, Santa Clara, California
Antal Rajnak, Mentor Graphics, Geneva, Switzerland
Françoise Simonot-Lion, Lorraine Laboratory of Computer Science Research and Applications (LORIA) Nancy, Vandoeuvre-lés-Nancy, France
Lothar Thiele, Swiss Federal Institute of Technology, Zürich, Switzerland
Tomas Weigert, Motorola, Schaumburg, Illinois
Reinhard Wilhelm, University of Saarland, Saarbrücken, Germany

I

Network Embedded Systems: An Introduction

1 Networked Embedded Systems: An Overview *Richard Zurawski* **1**-1
Networking of Embedded Systems • Automotive Networked Embedded Systems • Networks Embedded Systems in Industrial Automation • Wireless Sensor Networks • Networked Embedded Systems in Building Automation • Concluding Remarks

2 Middleware Design and Implementation for Networked Embedded Systems *Venkita Subramonian and Christopher D. Gill*..... **2**-1
Introduction • Middleware Solution Space • ORB Middleware for Networked Embedded Systems: A Case Study • Design Recommendations and Trade-Offs • Related Work • Concluding Remarks • Acknowledgments

1

Networked Embedded Systems: An Overview

1.1 Networking of Embedded Systems 1-1
1.2 Automotive Networked Embedded Systems 1-4
1.3 Networks Embedded Systems in Industrial Automation 1-7
1.4 Wireless Sensor Networks 1-9
1.5 Networked Embedded Systems in Building Automation 1-11
1.6 Concluding Remarks 1-13
References 1-13

Richard Zurawski
ISA Group

1.1 Networking of Embedded Systems

The last two decades have witnessed a remarkable evolution of embedded systems from being assembled from discrete components on printed circuit boards, although, they still are, to systems being assembled from IP components "dropped" on to silicon of the system on a chip. Systems on a chip offer a potential for embedding complex functionalities, and to meet demanding performance requirements of applications such as DSP, network, and multimedia processors. Another phase in this evolution, already in progress, is the emergence of distributed embedded systems; frequently termed as networked embedded systems, where the word "networked" signifies the importance of the networking infrastructure and communication protocol. A networked embedded system is a collection of spatially and functionally distributed embedded nodes interconnected by means of wireline and/or wireless communication infrastructure and protocols, interacting with the environment (via a sensor/actuator elements) and each other, and, possibly, a master node performing some control and coordination functions, to coordinate computing and communication to achieve certain goal(s). The networked embedded systems appear in a variety of application domains such as automotive, train, aircraft, office building, and industrial areas—primarily for monitoring and control, environment monitoring, and, in future, control, as well.

There have been various reasons for the emergence of networked embedded systems, influenced largely by their application domains. The benefits of using distributed systems and an evolutionary need to replace point-to-point wiring connections in these systems by a single bus are some of the most important ones.

The advances in design of embedded systems, tools availability, and falling fabrication costs of semiconductor devices and systems have allowed for infusion of intelligence into the field devices

such as sensors and actuators. The controllers used with these devices provide typically on-chip signal conversion, data and signal processing, and communication functions. The increased functionality, processing, and communication capabilities of controllers have been largely instrumental in the emergence of a widespread trend for networking of field devices around specialized networks, frequently referred to as field area networks.

The field area networks, or fieldbuses [1] (fieldbus is, in general, a digital, two-way, multidrop communication link) as commonly referred to, are, in general, the networks connecting field devices such as sensors and actuators with field controllers (for instance, programmable logic controllers (PLCs) in industrial automation, or electronic control units (ECUs) in automotive applications), as well as man–machine interfaces; for instance, dashboard displays in cars.

In general, the benefits of using those specialized networks are numerous, including increased flexibility attained through the combination of embedded hardware and software, improved system performance, and ease of system installation, upgrade, and maintenance. Specifically, in automotive and aircraft applications, for instance, they allow for a replacement of mechanical, hydraulic, and pneumatic systems by mechatronic systems, where mechanical or hydraulic components are typically confined to the end-effectors; just to mention this two different application areas.

Unlike local area networks (LANs), due to the nature of communication requirements imposed by applications, field area networks, by contrast, tend to have low data rates, small size of data packets, and typically require real-time capabilities which mandate determinism of data transfer. However, data rates above 10 Mbit/s, typical of LANs, have become a commonplace in field area networks.

The specialized networks tend to support various communication media like twisted pair cables, fiber-optic channels, power line communication, radio frequency channels, infrared connections, etc. Based on the physical media employed by the networks, they can be in general divided into three main groups, namely, wireline-based networks using media such as twisted pair cables, fiber-optic channels (in hazardous environments like chemical and petrochemical plants), and power lines (in building automation); wirelss networks supporting radio frequency channels and infrared connections; and hybrid networks, with wireline extended by wireless links [2].

Although the use of wireline-based field area networks is dominant, the wireless technology offers a range of incentives in a number of application areas. In industrial automation, for instance, wireless device (sensor/actuator) networks can provide a support for mobile operation required in case of mobile robots, monitoring and control of equipment in hazardous and difficult to access environments, etc. In a wireless sensor/actuator network, stations may interact with each other on a peer-to-peer basis, and with a base station. The base station may have its transceiver attached to a cable of a (wireline) field area network, giving rise to a hybrid wireless–wireline system [2]. A separate category is the wireless sensor networks, envisaged to be largely used for monitoring purposes.

The variety of application domains impose different functional and nonfunctional requirements on to the operation of networked embedded systems. Most of them are required to operate in a reactive way; for instance, systems used for control purposes. With that comes the requirement for real-time operation, in which systems are required to respond within a predefined period, mandated by the dynamics of the process under control. A response, in general, may be periodic to control a specific physical quantity by regulating dedicated end-effector(s), or aperiodic arising from unscheduled events such as out-of-bounds state of a physical parameter or any other kind of abnormal conditions. Broadly speaking, systems which can tolerate a delay in response are called soft real-time systems; in contrast, hard real-time systems require deterministic response to avoid changes in the system dynamics which potentially may have negative impact on the process under control, and as a result may lead to economic losses or cause injury to human operators. Representative examples of systems imposing hard real-time requirement on their operation are Fly-by-Wire in aircraft control, Steer-by-Wire in automotive applications, to mention some.

The need to guarantee a deterministic response mandates using appropriate scheduling schemes, which are frequently implemented in application domain-specific real-time operating systems or frequently custom designed "bare-bone" real-time executives.

The networked embedded systems used in safety-critical applications such as Fly-by-Wire and Steer-by-Wire require a high level of dependability to ensure that a system failure does not lead to a state in which human life, property, or environment are endangered. The dependability issue is critical for technology deployment; various solutions are discussed in this chapter in the context of automotive applications.

As opposed to applications mandating hard real-time operation, such as the majority of industrial automation controls or safety-critical automotive control applications, building automation control systems, for instance, seldom have a need for hard real-time communication; the timing requirements are much more relaxed. The building automation systems tend to have a hierarchical network structure and typically implement all seven layers of the ISO/OSI reference model [3]. In case of field area networks employed in industrial automation, for instance, there is little need for the routing functionality and end-to-end control. As a consequence, typically, only the layers 1 (physical layer), 2 (data link layer, including implicitly the medium access control layer), and 7 (application layer, which covers also user layer) are used in those networks.

This diversity of requirements imposed by different application domains (soft/hard real-time, safety critical, network topology, etc.) necessitated different solutions, and using different protocols based on different operation principles. This has resulted in a plethora of networks developed for different application domains.

Design methods for networked embedded systems fall into the general category of system-level design. They include three aspects, namely, node design (covered extensively in Section I of the book), network architecture design, and timing analysis of the whole system. The network architecture design involves a number of activities. One of them is selection of an appropriate communication protocol and communication medium. A safety-critical application will employ a protocol based on Time Division Multiple Access (TDMA) medium access control to ensure deterministic access to the medium. For an application in building automation and control, the choice of the communication medium may be the power line wires in the existing building or dedicated twisted pair wires in a new construction. The topology of the network heavily depends on the application area. In industrial automated systems, the prevalent topology is the bus. Building network may have a complex topology with many logical domains. Configuration of the communication protocol, among other things, involves allocation to the communication nodes priorities in the priority busses, or slots in the TDMA-based protocols, for instance. The timing analysis aims at obtaining actual times for the chosen architecture. That involves task execution time measures such as worst-case execution time (WCET), best-case execution time (BCET), and average execution time; response time of a task from invocation to completion; end-to-end delay; and jitter, or variation in execution time of a task, for instance. In the end, the whole system has to be schedulable to guarantee that deadlines of all distributed tasks communicating over the network will be met in all operational conditions the system is anticipated to be subjected to. As an example, let us consider a simple control loop comprising a sensing node with a single application task dedicated to sensing, an actuator node processing data received from the sensing node, and generating control value delivered to an actuator over a dedicated link. The composite time of data processing (WCET) and transmission (worst-case response time) has to be shorter or equal to the maximum time allowed by the process dynamics under control. In case of other nodes connected to the shared communication network and forming similar control loops, a contention for the medium access may arise to be remedied for safety-critical and hard real-time systems by adopting a fixed transmission schedule as in the case of the time-triggered TDMA-based protocols, for instance. The schedulability analysis is to determine if the worst-case response time for all those composite tasks forming control loops is less then or equal to the deadline.

1.2 Automotive Networked Embedded Systems

Trends for networking also emerged in the automotive electronic systems where the ECUs are networked by means of one of automotive-specific communication protocols for the purpose of controlling one of the vehicle functions; for instance, electronic engine control, antilocking break system, active suspension, telematics, and to mention a few. In Ref. [4], a number of functional domains have been identified for the deployment of automotive networked embedded systems. They include the power train domain, involving, in general, control of engine and transmission; the chassis domain involving control of suspension, steering, and braking, etc.; the body domain involving control of wipers, lights, doors, windows, seats, mirrors, etc.; the telematics domain involving mostly the integration of wireless communications, vehicle monitoring systems, and vehicle location systems; and the multimedia and human–machine interface domains. The different domains impose varying constraints on the networked embedded systems in terms of performance, safety requirements, or Quality of Services (QoS). For instance, the power train and chassis domains will mandate real-time control; typically bounded delay is required, as well as fault-tolerant services.

There are a number of reasons for the interest of the automotive industry in adopting mechatronic solutions, known by their generic name as X-by-Wire, aiming to replace mechanical, hydraulic, and pneumatic systems by electrical/electronic systems. The main factors seem to be economic in nature, improved reliability of components, and increased functionality to be achieved with a combination of embedded hardware and software. Steer-by-Wire, Brake-by-Wire, or Throttle-by-Wire systems are representative examples of those systems. But, it seems that certain safety-critical systems such as Steer-by-Wire and Brake-by-Wire will be complemented with traditional mechanical/hydraulic backups, for safety reasons. The dependability of X-by-Wire systems is one of the main requirements, as well as constraints on the adoption of this kind of systems. In this context, a safety-critical X-by-Wire system has to ensure that a system failure does not lead to a state in which human life, property, or environment is endangered; and a single failure of one component does not lead to a failure of the whole X-by-Wire system [5]. When using Safety Integrity Level scale, it is required for X-by-Wire systems that the probability of a failure of a safety-critical system does not exceed the figure of 10^{-9} per hour/system. This figure corresponds to the SIL4 level. Another equally important requirement for the X-by-Wire systems is to observe hard real-time constraints imposed by the system dynamics; the end-to-end response times must be bounded for safety-critical systems. A violation of this requirement may lead to performance degradation of the control system, and other consequences as a result. Not all automotive electronic systems are safety critical. For instance, system(s) to control seats, door locks, internal lights, etc. are not. Different performance, safety, and QoS requirements dictated by various in-car application domains necessitate adoption of different solutions, which, in turn, gave rise to a significant number of communication protocols for automotive applications. Time-triggered protocols (TTP) based on TDMA medium access control technology are particularly well suited for the safety-critical solutions, as they provide deterministic access to the medium. In this category, there are two protocols, which, in principle, meet the requirements of X-by-Wire applications, namely, TTP/C [6] and FlexRay [7] (FlexRay can support a combination of both time-triggered and event-triggered transmissions). The following discussion will focus mostly on TTP/C and FlexRay.

The TTP/C is a fault-tolerant TTP; one of two protocols in the time-triggered architecture (TTA) [8]. The other one is a low-cost fieldbus protocol TTP/A [9]. In TTA, the nodes are connected by two replicated communication channels forming a cluster. In TTA, a network may have two different interconnection topologies, namely, bus and star. In the bus configuration, each node is connected to two replicated passive buses via bus guardians. The bus guardians are independent units preventing associated nodes from transmitting outside predetermined time slots, by blocking the transmission path; a good example may be a case of a controller with a faulty clock oscillator, which attempts to transmit continuously. In the star topology, the guardians are integrated into two replicated central star couplers. The guardians are required to be equipped with their own clocks, distributed clock

synchronization mechanism, and power supply. In addition, they should be located at a distance from the protected node to increase immunity to spatial proximity faults. To cope with internal physical faults, TTA employs partitioning of nodes into so-called fault-tolerant units (FTUs), each of which is a collection of several stations performing the same computational functions. As each node is (statically) allocated a transmission slot in a TDMA round, failure of any node, or a frame corruption is not going to cause degradation of the service. In addition, data redundancy allows, by voting process, to ascertain the correct data value.

TTP/C employs synchronous TDMA medium access control scheme on replicated channels, which ensures fault-tolerant transmission with known delay and bounded jitter between the nodes of a cluster. The use of replicated channels, and redundant transmission, allows for the masking of a temporary fault on one of channels. The payload section of the message frame contains up to 240 bytes of data protected by a 24-bit CRC checksum. In TTP/C the communication is organized into rounds. In a round, different slot sizes may be allocated to different stations. However, slots belonging to the same station are of the same size in successive rounds. Every node must send a message in every round. Another feature of TTP/C is fault-tolerant clock synchronization that establishes global time base without a need for a central time provider. In the cluster, each node contains the message schedule. Based on that information, a node computes the difference between the predetermined and actual arrival time of a correct message. Those differences are averaged by a fault-tolerant algorithm which allows for the adjustment of the local clock to keep it in synchrony with clocks of other nodes in the cluster. TTP/C provides so-called membership service to inform every node about the state of every other node in the cluster; it is also used to implement the fault-tolerant clock synchronization mechanism. This service is based on a distributed agreement mechanism which identifies nodes with failed links. A node with a transmission fault is excluded from the membership until restarted with a proper state of the protocol. Another important feature of TTP/C is a clique avoidance algorithm to detect and eliminate formation of cliques in case the fault hypothesis is violated. In general, the fault-tolerant operation based on FTUs cannot be maintained if the fault hypothesis is violated. In such a situation, TTA activates never-give-up (NGU) strategy [9]. The NGU strategy, specific to the application, is initiated by TTP/C in combination with the application with an aim to continue operation in a degraded mode.

The TTA infrastructure and the TTP/A and TTP/C protocols have a long history dating back to 1979 when the Maintainable Architecture for Real-Time Systems project started at the Technical University of Berlin. Subsequently, the work was carried out at the Vienna University of Technology. TTP/C protocols have been experimented with and considered for deployment for quite some time. However, to date, there have been no actual implementations of that protocol involving safety-critical systems in commercial automobiles, or trucks. In 1995, a "proof of concept," organized jointly by Vienna University of Technology and DaimlerChrysler, demonstrated a car equipped with a "Brake-by-Wire" system based on time-triggered protocol.

FlexRay, which appears to be the frontrunner for future automotive safety-critical control applications, employs a modified TDMA medium access control scheme on a single or replicated channel. The payload section of a frame contains up to 254 bytes of data protected by a 24-bit CRC checksum. To cope with transient faults, FlexRay also allows for a redundant data transmission over the same channel(s) with a time delay between transmissions. The FlexRay communication cycle comprises of a network communication time and network idle time. Two or more communication cycles can form an application cycle. The network communication time is a sequence of static segment, dynamic segment, and symbol window. The static segment uses a TDMA MAC protocol. The static segment comprises of static slots of fixed duration. Unlike in TTP/C, the static allocation of slots to a node (communication controller) applies to one channel only. The same slot may be used by another node on the other channel. Also, a node may possess several slots in a static segment. The dynamic segment uses a Flexible Time Division Multiple Access (FTDMA) MAC protocol, which allows for a priority and demand-driven access pattern. The dynamic segment comprises of so-called mini-slots

with each node allocated a certain number of mini-slots, which do not have to be consecutive. The mini-slots are of a fixed length, and much shorter than static slots. As the length of a mini-slot is not sufficient to accommodate a frame (a mini-slot only defines a potential start time of a transmission in the dynamic segment), it has to be enlarged to accommodate transmission of a frame. This in turn reduces the number of mini-slots in the reminder of the dynamic segment. A mini-slot remains silent if there is nothing to transmit. The nodes allocated mini-slots toward the end of the dynamic segment are less likely to get transmission time. This in turn enforces a priority scheme. The symbol window is a time slot of fixed duration used for network management purposes. The network idle time is a protocol-specific time window in which no traffic is scheduled on the communication channel. It is used by the communication controllers for the clock synchronization activity; in principle, similar to the one described for TTP/C. If the dynamic segment and idle window are optional, the idle time and minimal static segment are mandatory parts of a communication cycle; minimum two static slots (degraded static segment) or four static slots for fault-tolerant clock synchronization are required. With all that, FlexRay allows for three configurations: pure static; mixed, with both static and dynamic—bandwidth ratio depends on the application; and pure dynamic, where all bandwidth is allocated to the dynamic communication.

FlexRay supports a range of network topologies offering a maximum of scalability and a considerable flexibility in the arrangement of embedded electronic architectures in automotive applications. The supported configurations include bus, active star, active cascaded stars, and active stars with bus extension. FlexRay also uses the bus guardians in the same way as TTP/C.

The existing FlexRay communication controllers support communication bit rates of up to 10 Mbps on two channels. The transceiver component of the communication controller also provides a set of automotive network-specific services. Two major services are alarm handling and wake-up control. In addition to the alarm information received in a frame, an ECU also receives the alarm symbol from the communication controller. This redundancy can be used to validate critical signals; for instance, an air-bag fire command. The wake-up service is required where electronic components have a sleep mode to reduce power consumption.

FlexRay is a joint effort of a consortium involving some of the leading car makers and technology providers, to mention BMW, Bosch, Daimler, Freescale Semiconductor, General Motors, NXP Semiconductors, Volkswagen as the core partners.

Time-Triggered Controller Area Networks (TTCAN) [10], which can support a combination of both time- and event-triggered transmissions, utilize physical and Data-Link Layer of the Controller Area Network (CAN) protocol. As this protocol, as in the standard, does not provide necessary dependability services, it is unlikely to play any role in fault-tolerant communication in automotive applications.

TTP/C and FlexRay protocols belong to class D networks in the classification published by the Society for Automotive Engineers [11,12]. Although the classification dates back to 1994, it is still a reasonable guideline for distinction of different protocols based on data transmission speed and functions distributed over the network. The classification comprises of four classes. Class A includes networks with a data rate less than 10 Kbit/s. Some of the representative protocols are Local Interconnect Network (LIN) [13] and TTP/A [9]. Class A networks are employed largely to implement the body domain functions. Class B networks operate within the range of 10 to 125 Kbit/s. Some of the representative protocols are J1850 [14], low-speed CAN [15], and Vehicle Area Network (VAN) [16]. Class C networks operate within the range of 125 Kbit/s to 1 Mbit/s. Examples of this class networks are high-speed CAN [17] and J1939 [18]. Network in this class are used for the control of power train and chassis domains. High-speed CAN, although used in the control of power train and chassis domains, is not suitable for safety-critical applications as it lacks the necessary fault-tolerant services. Class D networks (not formally defined as yet) includes networks with a data rate over 1 Mb/s. Networks to support the X-by-Wire solutions fall into this class, to include TTP/C and FlexRay. Also, Media

Oriented System Transport (MOST) [19] and IDB-1394 [20] for multimedia applications belong the this class.

The cooperative development process of networked embedded automotive applications brings with itself heterogeneity of software and hardware components. Even with the inevitable standardization of those components, interfaces, and even complete system architectures, the support for reuse of hardware and software components is limited, thus potentially making the design of networked embedded automotive applications labor intensive, error-prone, and expensive. This situation spurned a number of standardization initiatives aimed at the development of component-based design integration methodologies of which OSEK/VDX [21] and AUTomotive Open System ARchitecture (AUTOSAR) [22] are the most notable (both discussed in detail in this book).

1.3 Networks Embedded Systems in Industrial Automation

Although for the origins of field area networks, one can look back as far as the end of 1960s in the nuclear instrumentation domain, CAMAC network [23], and the beginning of 1970s in avionics and aerospace applications, MIL-STD-1553 bus [24], it was the industrial automation area which brought the main thrust of developments. The need for integration of heterogeneous systems, difficult at the time due to the lack of standards, resulted in two major initiatives that have had a lasting impact on the integration concepts, and architecture of the protocol stack of field area networks. These initiatives were Technical and Office Protocol (TOP) [25] and Manufacturing Automation Protocol (MAP) [26] projects. The two projects exposed some pitfalls of the full seven-layer stack implementations (ISO/OSI model [3]). As a result, typically, only the layers 1 (physical layer), 2 (data link layer, including implicitly the medium access control layer), and 7 (application layer, which covers also user layer) are used in the field area networks [27]; also prescribed by the international fieldbus standard, IEC 61158 [28]. In IEC 61158, functions of layer 3 and 4 are recommended to be placed either in layer 2 or layer 7—network and transport layers are not required in a single segment network typical of process and industrial automation (situation is different though in building automation, for instance, where the routing functionality and end-to-end control may be needed arising from a hierarchical network structure); functions of layers 5 and 6 are always covered in layer 7.

The evolution of fieldbus technology which begun well over two decades ago has resulted in a multitude of solutions reflecting the competing commercial interests of their developers and standardization bodies, both national and international: IEC [29], ISO [30], ISA [31], CENELEC [32], and CEN [33]. This is also reflected in IEC 61158 (adopted in 2000), which accommodates all national standards and user organization championed fieldbus systems. Subsequently, implementation guidelines were compiled in to Communication Profiles, IEC 61784-1 [34]. Those Communication Profiles identify seven main systems (or Communication Profile Families) known by brand names as Foundation Fieldbus (H1, HSE, H2) used in process and factory automation; ControlNet and EtherNet/IP both used in factory automation; PROFIBUS (DP, PA) used in factory and process automation, respectively; PROFINET used in factory automation; P-Net (RS 485, RS 232) used in factory automation and shipbuilding; WorldFIP used in factory automation; INTERBUS, INTERBUS TCP/IP, and INTERBUS Subset used in factory automation; Swiftnet "transport," Swiftnet "full stack" used by aircraft manufacturers. The listed application areas are the dominant ones.

Ethernet, the backbone technology for office networks, is increasingly being adopted for communication in factories and plants at the fieldbus level. The random and native CSMA/CD arbitration mechanism is being replaced by other solutions allowing for deterministic behavior required in real-time communication to support soft and hard real-time deadlines, time synchronization of activities required to control drives, for instance, and for exchange of small data records characteristic of monitoring and control actions. The emerging Real-Time Ethernet (RTE), Ethernet augmented

with real-time extensions, under standardization by IEC/SC65C committee, is a fieldbus technology which incorporates Ethernet for the lower two layers in the OSI model. Additional profiles for ISO/IEC 8802.3 (Ethernet) based communication networks in real-time applications were defined in the IEC 61784-2 standard [35]. There are already a number of implementations, which use one of the three different approaches to meet real-time requirements. First approach is based on retaining the TCP/UDP/IP protocols suite unchanged (subject to nondeterministic delays), all real-time modifications are enforced in the top layer. Implementations in this category include Modbus/TPC (Profile 15/1 & 15/2) [36] (defined by Schneider Electric and supported by Modbus-IDA [37]), EtherNet/IP (Profile 2/2 & 2/2.1) [38] (defined by Rockwell and supported by the Open DeviceNet Vendor Association (ODVA) [39] and ControlNet International [40]), P-Net (Profile 4/3) [41] (proposed by the Danish P-Net national committee), and Vnet/IP (Profile 10/1) (developed by Yokogawa, Japan [42]). In the second approach, the TCP/UDP/IP protocols suite is bypassed, the Ethernet functionality is accessed directly—in this case, RTE protocols use their own protocol stack in addition to the standard IP protocol stack. The implementations in this category include Ethernet Powerlink (EPL) (Profile 13/1) (defined by Bernecker + Rainer (B&R), and now supported by the EPL Standardisation Group [43]), TCnet (Profile 11/1) (a Time-critical Control Network—a proposal from Toshiba), EPA (Profile 14/1 & 14/2) (Ethernet for Plant Automation) (a Chinese proposal), and PROFIBUS CBA (Profile 3/3) (Component-Based Automation) [44] (defined by several manufacturers including Siemens, and supported by PROFIBUS International [45]). Finally, in the third approach, the Ethernet mechanism and infrastructure are modified. The implementations include SERCOS III (Profile 16/3) [46] (under development by SERCOS), EtherCAT (Profile 12/1 & 12/2) [47] (defined by Beckhoff and supported by the EtherCAT Technology Group [48]), and PROFINET IO (Profile 3/4, 3/5, 3/6) (defined by several manufacturers including Siemens, and supported by PROFIBUS International).

The use of standard components such as protocol stacks, Ethernet controllers, bridges, etc. allows to mitigate the ownership and maintenance cost. The direct support for the Internet technologies allows for vertical integration of various levels of industrial enterprise hierarchy to include seamless integration between automation and business logistic levels to exchange jobs and production (process) data; transparent data interfaces for all stages of the plant life cycle; the Internet- and Web-enabled remote diagnostics and maintenance, as well as electronic orders and transactions. In case of industrial automation, the advent and use of networking have allowed for horizontal and vertical integration of industrial enterprises.

With the growing trend for networking of embedded system and their internetworking with LAN, WAN, and the Internet (for instance, there is a growing demand for remote access to process data at the factory floor), many of those systems may become exposed to potential security attacks, which may compromise their integrity and cause damage as a result. Potential security solutions for this kind of systems depend heavily on the specific device or system protected, application domain, and extent of internetworking and its architecture.

Typically, office IT operational security requirements involve confidentiality to protect data from unauthorized entities, integrity to protect against unauthorized data manipulation, and availability to ensure data are available when needed. Operational security requirements for automation and process control systems, unlike in the office IT, focus on safety to guarantee absence of catastrophic consequences for humans and environment, and system/plant availability—the automation system and plant have to be safe operational over extended periods, even if they continue operation in a degraded mode in the presence of a fault caused by a security attack. Electronic security attacks may compromise the integrity of these systems and endanger plant, personnel, and even public safety. Unavailability may result in financial losses.

The security measures to be taken at the corporate and control network levels in the automated plant control hierarchy are essentially the same as in general networking. The situation is though different at the field and device level. At the field level, fieldbuses, in general, do not have any embedded

security features. As they are frequently located at the premises requiring access permit, eavesdropping or message tampering would require a physical access to the medium. A potential solution to provide a certain level of security is the access point (PLC, for instance) control. The emerging Ethernet-based fieldbuses are more vulnerable to attacks on account of using the Ethernet and the TCP/IP protocols and services. Here, the general communication security tools for TCP/IP apply.

At the device and embedded level, the limited computing, memory, and communication bandwidth resources of controllers embedded in the field devices pose considerable challenge for the implementation of effective security policies which, in general, are resource demanding. This limits the applicability of the mainstream cryptographic protocols, even vendor-tailored versions. The operating systems running on small footprint controllers tend to implement essential services only, and do not provide authentication or access control to protect mission and safety-critical field devices. In applications restricted to the Hypertext Transfer Protocol (HTTP), such as embedded Web servers, Digest Access Authentication [49], a security extension to HTTP, may offer an alternative and viable solution. In case of the denial of service (DoS) attack, the processor is preoccupied with handling communication interrupts potentially compromising the (hard) real-time requirements and operational safety as a result—clever interrupt priority allocation and/or selection are needed. Interrupts handling outside the normal operational conditions can cause with time battery draining on battery powered devices, making embedded controllers to become unavailable—a serious problem in wireless sensor networks deployed on the factory floor where the function of the unavailable node cannot be taken over by other nodes. In general, an embedded controller is expected to withstand autonomously many security attacks; the "buffer overflow" attack, for instance, which has the potential to crash the system—proper error and exception handling is required here.

1.4 Wireless Sensor Networks

Another trend in networking of field devices has emerged recently, namely, wireless sensor networks, which is another example of networked embedded systems. Here, the "embedded" factor is not so evident as in other applications; particularly true for the ad-hoc and self-organizing networks where the nodes may be embedded in the ecosystem or a battlefield, and to mention some.

Although reports on actual applications are scarce, and potential applications in the projected areas are still under consideration, the wireless sensor/actuator networks are in the deployment stage by the manufacturing industry. The use of wireless links with field devices, such as sensors and actuators, allows for flexible installation and maintenance, mobile operation required in case of mobile robots, and alleviates problems with cabling. A wireless communication system to operate effectively in the industrial/factory floor environment has to guarantee high reliability, low and predictable delay of data transfer (typically, less than 10 ms for real-time applications), support for high number of sensor/actuators (over 100 in a cell of a few meters radius), and low power consumption, and to mention some. In the industrial environments, the characteristic for the wireless channel degradation artifacts can be compounded by the presence of electric motors or a variety of equipments causing the electric discharge, which contribute to even greater levels of bit error and packet losses. One way to partially alleviate the problem is either by designing robust and loss-tolerant applications and control algorithms, or by trying to improve the channel quality; all subject of extensive research and development.

There is quite a difference in the requirements between the self-organizing and ad-hoc wireless sensor networks and those imposed by the industrial applications. Some of the major characteristics of the self-organizing and ad-hoc wireless sensor networks involve self-containment, lack of prearranged network topology (organized by nodes on ah-hoc basis), and the ability to self-heal (network

operation not affected if a node goes down). On the other hand, the wireless sensor networks in industrial applications feature a prearranged network topology; have no ability to self-heal; require long lifetime—both nodes and the network; and tend to be expensive. The prearranged network topology is determined by the discrete manufacturing or continuous process equipment arrangement, or system architecture, which determines where the information is to be collected. A network and system operation is affected if a node goes down; a physical process parameter such as temperature or pressure is no longer observable and controllable, for instance. A solution here is node redundancy, though expensive. The nodes are required to have a long lifetime mandated by economics of the production process and the initial investments. Node or node's components replacement can only by economically justified during the device-or system-scheduled or preemptive maintenance period. Fault-tolerance, physical, and operational robustness are some of the requirements imposed on the nodes. The required long lifetime also applies to batteries, if the node is battery powered. One of the consequences of the long lifetime requirement is the increased node functionality to provide a certain level of redundancy on chip or on printed circuit board assembly which comes at a cost. Another factor to have an impact on the cost is the IC packaging and node housing to protect the node from the harmful environmental factors such as temperature, fluids, and to mention some, which have a potential to shorten the operational lifetime of the node—or to prevent any electrical discharge from the node in hazardous environments, presence of flammable gases, for instance.

The operation of wireless sensor networks in industrial applications typically imposes some sort of real-time restrictions, and frequently hard bounds on the maximum delay. In most of "classical" wireless sensor networks, the time a packet takes to travel to its destination tends not to be an issue. This latency is typically tens of milliseconds for discrete manufacturing, tens of seconds for process control, and minutes to hours for the assets management.

The wireless sensor networks in industrial applications tend to have a hybrid wireless–wireline architecture interconnected via a gateway device, where high capacity and mains powered wireline is used for data distribution from the collection point [2]. In this context, the most established network topology is the star topology with wireless links between the sensor nodes and the gateway. The one-hop communication with the gateway makes this solution suitable for time-bounded communication supporting processes with fast dynamics. The hybrid mesh topology, another commonly used topology, has at its center the gateway device connected by wireless links to frequently mains powered router nodes, with the sensor nodes communication in one-hop communication with the router nodes.

Reliability of a message delivery is an important requirement in wireless sensor networks used in industrial applications. Lost messages may adversely change dynamics of a process under control, for instance. One way to improve the reliability is through transmission redundancy. Depending on the application, a number of options can be considered: space diversity—transmission through different paths; frequency diversity—on different frequencies; time diversity—several times on the same frequency; and modulation scheme diversity—different modulation schemes.

With nodes frequently located in inaccessible and/or hazardous places, combined with prohibitive cost of unscheduled energy source replacement requiring halting operational activities, low power consumption is of paramount importance in the wireless sensor networks used in industrial applications. Some of the factors minimizing power consumption include: using low power elements—CPU, for instance, runs on reduced clock rate with less on-chip functionality; selecting proper operational regime—adopting sleep/wake-up mode of operation, with transmission activated only if the value of a measured physical quantity is larger then the predetermined bound; choice of the right communication protocol; etc.

The communication protocol ultimately dictates the lower bound on the power consumption. The Wireless Interface to Sensors and Actuators (WISA) protocol [50,51] is a low power and high performance protocol based on single-hop transmission to avoid delays in intermediate nodes and Time Division Multiplexing (TDM) assuring no collisions to occur (a node is alone on the

channel). WISA is suitable for discrete manufacturing if the single-hop requirement is met (the star topology). Another option is the IEEE 802.15.4/ZigBee [52] technology. ZigBee supports multihop transmission mode of operation which requires the intermediate nodes (typically router nodes) to be mains powered. Unlike TDMA-based WISA, there is no allocation of timeslots to messages giving rise to contention for channel access, increased latency, and power consumption. It is suitable for applications with a relatively slow process dynamics such as process control, asset monitoring, etc.

The two most notable standardization initiatives aiming at wireless sensor networks for industrial applications are WirelessHART [53] and ISA100 [54]. WirelessHART, which extends the HART standard into the wireless domain, is the only standard available to cover wireless sensor networks for industrial applications. ISA100, still in the standardization process, aims at supporting different fieldbus formats, including HART, and applications ranging from safety-critical control to monitoring.

In practical implementations of wireless sensor networks in industrial applications, to leverage low cost, small size, and low power consumptions, standard Bluetooth (IEEE 802.15.1) 2.4 GHz radio transceivers [55,56] may be used as the sensor/actuators communication hardware. To meet the requirements for high reliability, low and predictable delay of data transfer, and support for high number of sensor/actuators, custom-optimized communication protocols may be required as the commercially available solutions such as Bluetooth (IEEE 802.15.1), IEEE 802.15.4 [52], and IEEE 802.11 [57–59] variants may not fulfill all the requirements. A representative example of this kind of systems is a wireless sensor/actuator network developed by ABB and deployed in a manufacturing environment [60]. The system, known as WISA (wireless sensor/actuator) has been implemented in a manufacturing cell to network proximity switches, which are some of the most widely used position sensors in automated factories to control positions of a variety of equipment, including robotic arms, for instance. The sensor/actuators communication hardware is based on a standard Bluetooth 2.4 GHz radio transceiver and low power electronics that handle the wireless communication link. The sensors communicate with a wireless base station via antennas mounted in the cell. For the base station, a specialized RF front end was developed to provide collision-free air access by allocating a fixed TDMA time slot to each sensor/actuator. Frequency hopping (FH) was employed to counter both frequency-selective fading and interference effects, and operates in combination with automatic retransmission requests. The parameters of this TDMA/FH scheme were chosen to satisfy the requirements of up to 120 sensor/actuators per base station. Each wireless node has a response or cycle time of 2 ms, to make full use of the available radio band of 80 MHz width. The FH sequences are cell-specific and were chosen to have low cross-correlations to permit parallel operation of many cells on the same factory floor with low self-interference. The base station can handle up to 120 wireless sensor/actuators and is connected to the control system via a (wireline) fieldbus. To increase capacity, a number of base stations can operate in the same area. WISA provides wireless power supply to the sensors, based on magnetic coupling.

1.5 Networked Embedded Systems in Building Automation

Another fast growing application area for networked embedded systems is building automation [61]. Building automation systems aim at the control of the internal environment, as well as the immediate external environment of a building, or building complex. At present, the focus of research and technology development is on commercial type of buildings (office building, exhibition centre, shopping complex, etc.). In future, this will also include industrial type of buildings, which pose substantial challenges to the development of effective monitoring and control solutions. Some of the main services to be offered by the building automation systems typically include climate control to include heating, ventilation, air conditioning; visual comfort to cover artificial lighting, control of day light; safety services such as fire alarm, and emergency sound system; security protection; control of

utilities such as power, gas, water supply, etc.; internal transportation systems to mention lifts, escalators, etc.

In terms of the QoS requirements imposed on the field area networks, building automation systems differ considerably from their counterparts in industrial automation, for instance. There is seldom a need for hard real-time communication; the timing requirements are much more relaxed. Traffic volume in normal operation is low. Typical traffic is event driven, and mostly uses peer-to-peer communication paradigm. Fault-tolerance and network management are important aspects. As with industrial fieldbus systems, there are a number of bodies involved in the standardization of technologies for building automation, including the field area networks. The communication architecture supporting automation systems embedded in the buildings has typically three levels: field, control, and management levels. The field level involves operation of elements such as switches, motors, lighting cells, dry cells, etc. The peer-to-peer communication is perhaps most evident at that level; toggling a switch should activate a lighting cell(s), for instance. The automation level is typically used to evaluate new control strategies for the lower level in response to the changes in the environment; reduction in the day light intensity, external temperature change, etc. LonWorks [62], BACnet [63], and EIB/KNX [64–67] are open system networks, which can be used at more than one level of the communication architecture. A round up of LonWorks will be provided in the following, as the most widely used in building automation specialized field area network.

LonWorks (EIA-709), a trademark of Echelon Corp. [68], employs LonTalk protocol which implements all seven layers of the ISO/OSI reference model. The LonTalk protocol was published as a formal standard [69], and revised in 2002 [70].

In EIA-709, layer 2 supports various communication media-like twisted pair cables (78 kbps (EIA-709.3) or 1.25 Mbps), power line communication (4 kbps, EIA-709.2), radio frequency channel, infrared connections, fiber-optic channels (1.25 Mbps), as well as IP connections based on the EIA-852 protocol standard [71] to tunnel EIA-709 data packets through IP (Intranet, Internet) networks. A p-persistent CSMA bus arbitration scheme is used on twisted pair cables. For other communication media, the EIA-709 protocol stack uses the arbitration scheme defined for the very media.

The EIA-709 layer 3 supports a variety of different addressing schemes and advanced routing capabilities. The entire routable address space of a LonTalk network is referred to as the domain. A domain is restricted to 255 subnets; a subnet allows for up to 127 nodes. The total number of addressable nodes in a domain can reach 32385; up to 2^{48} domains can be addressed. Domain gateways can be built between logical domains to allow communication across domain boundaries. Groups can be formed to send a single data packet to a group of nodes using a multicast-addressed message. Routing is performed between different subnets only. An EIA-709 node can send a unicast-addressed message to exactly one node using either a unique 48-bit node identification *(Node ID)* address or the logical subnet/node address. A multicast addressed message can be sent to either a group of nodes (group address), or to all nodes in the subnet, or all nodes in the entire domain (broadcast address).

The EIA-709 layer 4 supports four types of services. The unacknowledged service transmits the data packet from the sender to the receiver. The unacknowledged repeated service transmits the same data packet a number of times. The number of retries is programmable. The acknowledged service transmits the data packet and waits for an acknowledgement from the receiver. If not received by the transmitter, the same data packet is sent again. The number of retries is programmable. The requested response service sends a request message to the receiver; the receiver must respond with a response message, for instance, with statistics information. There is a provision for authentication of acknowledged transmissions, although not very efficient.

Network nodes (which, typically, include Neuron chip, RAM/Flash, power source clock, network transceiver, and input/output interface connecting to sensor and actuator) can be based on the Echelon's Neuron chip series manufactured by Motorola, Toshiba, an Cypress; recently also based on other platform-independent implementations such a LoyTec LC3020 controller. The Neuron chips-based

controllers are programmed with the Echelon's Neuron C language, which is a derivative of ANSI C. Other controllers such as LC3020 are programmed with standard ANSI C. The basic element of Neuron C is the network variable (NV), which can be propagated over the network. For instance, SNVT_temp represents temperature in degree Celsius; SNVT stands for Standard NV Type. Network nodes communicate with each other by exchanging NVs. Another way to communicate between nodes is by using explicit messages. The Neuron C programs are used to schedule application events and to react to incoming data packets (receiving NV) from the network interface. Depending on the network media and the network transceivers, a variety of network topologies are possible with LonWorks nodes, to include bus, ring, star, and free topology.

As the interoperability on all seven OSI layers does not guarantee interworkable products, the LonMark organization [72] has published interoperability guidelines for nodes that use the LonTalk protocol. A number of task groups within LonMark define functional profiles (subset of all the possible protocol features) for analog input, analog output, temperature sensor, etc. The task groups focus on various types of applications such as home/utility, HVAC, lighting, etc.

LonBuilder and NodeBuilder are development and integration tools offered by Echelon. Both tools allow writing Neuron C programs, to compile and link them and download the final application into the target node hardware. NodeBuilder supports debugging of one node at the time. LonBuilder, which supports simultaneous debugging of multiple nodes, has a built-in protocol analyzer and a network binder to create communication relationships between network nodes. The Echelon's LNS (network operating system) provides tools that allow one to install, monitor, control, manage, and maintain control devices, and to transparently perform these services over any IP-based network, including the Internet.

1.6 Concluding Remarks

This chapter has presented an overview of trends for networking of embedded systems, their design, and selected application domain-specific network technologies. The networked embedded systems appear in a variety of application domains to mention automotive, train, aircraft, office building, and industrial automation. With the exception of building automation, the systems discussed in this chapter tend to be confined to a relatively small area covered and limited number of nodes, as in case of an industrial process, an automobile, or a truck. In the building automation controls, the networked embedded systems may take on truly large proportions in terms of area covered and number of nodes. For instance, in a LonTalk network, the total number of addressable nodes in a domain can reach 32385; up to 2^{48} domains can be addressed.

The wireless sensor/actuator networks, as well as wireless–wireline hybrid networks, have started evolving from the concept to actual implementations, and are poised to have a major impact on industrial, home, and building automation—at least in these application domains, for a start.

The networked embedded systems pose a multitude of challenges in their design, particularly for safety-critical applications, deployment, and maintenance. The majority of the development environments and tools for specific networking technologies do not have firm foundations in computer science, software engineering models, and practices making the development process labor-intensive, error-prone, and expensive.

References

1. R. Zurawski, The industrial communication systems, *Proceedings of the IEEE*, 93(6):1067–1072, 2005.
2. J.-D. Decotignie, P. Dallemagne, and A. El-Hoiydi, Architectures for the interconnection of wireless and wireline fieldbusses, *Proceedings of the 4th IFAC Conference on Fieldbus Systems and Their Applications 2001 (FET 2001)*, Nancy, France, 2001.

3. H. Zimmermann, OSI reference model: The ISO model of architecture for open system interconnection, *IEEE Transactions on Communications*, 28(4):425–432, 1980.
4. N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, Trends in automotive communication systems, *Proceedings of the IEEE*, 93(6):1204–1223, 2005.
5. X-by-Wire Project, Brite-EuRam 111 Program, X-by-Wire—Safety Related Fault Tolerant Systems in Vehicles, Final Report, 1998.
6. TTTech Computertechnik GmbH. *Time-Triggered Protocol TTP/C*, High-Level Specification Document, Protocol Version 1.1, November 2003. [Online]. www.tttech.com
7. FlexRay Consortium, *FlexRay Communication System, Protocol Specification, Version 2.1*, December 2005. [Online]. www.flexray.com
8. H. Kopetz and G. Bauer, The time triggered architecture, *Proceedings of the IEEE*, 91(1):112–126, 2003.
9. H. Kopetz, et al., *Specification of the TTP/A Protocol*, University of Technology Vienna, Vienna, September 2002.
10. International Standard Organization, *11898–4, Road Vehicles—Controller Area Network (CAN)—Part 4: Time-Triggered Communication*, ISO, 2000.
11. Society of Automotive Engineers, J2056/1 class C application requirements classifications, *SAE Handbook*, SAE Press, Warrendale, PA, 1994.
12. Society of Automotive Engineers, J2056/2 survey of known protocols, *SAE Handbook*, Vol. 2, SAE Press, Warrendale, PA, 1994.
13. A. Rajnak, The LIN standard, *The Industrial Communication Technology Handbook*, CRC Press, Boca Raton, FL, 2005.
14. Society of Automotive Engineers, *Class B Data Communications Network Interface—SAE J1850 Standard—Rev. Nov. 96*, 1996.
15. International Standard Organization, *ISO 11519–2, Road Vehicles—Low Speed Serial Data Communication—Part 2: Low Speed Controller Area Network, ISO*, 1994.
16. International Standard Organization, *ISO 11519–3, Road Vehicles—Low Speed Serial Data Communication—Part 3: Vehicle Area Network (VAN), ISO*, 1994.
17. International Standard Organization, *ISO 11898, Road Vehicles—Interchange of Digital Information—Controller Area Network for High-Speed Communication, ISO*, 1994.
18. SAE J1939 Standards Collection. [Online]. www.sae.org
19. MOST Cooperation, *MOST Specification Revision 2.3*, August 2004. [Online]. www.mostnet.de
20. [Online]. www.idbforum.org
21. OSEK Consortium [Online]. www.osek-vdx.org
22. AUTOSAR Consortium [Online]. www.autosar.org
23. L. Costrell, CAMAC instrumentation system—Introduction and general description. *IEEE Transactions on Nuclear Science*, 18(2):3–8, 1971.
24. C.-A. Gifford, A military standard for multiplex data bus, *Proceedings of the IEEE-1974, National Aerospace and Electronics Conference*, Dayton, OH, May 13–15, 1974, pp. 85–88.
25. N. Collins, Boeing architecture and TOP (technical and office protocol), *Networking: A Large Organization Perspective*, Melbourne, FL, April 1986, pp. 49–54.
26. H. A. Schutz, The role of MAP in factory integration. *IEEE Transactions on Industrial Electronics*, 35(1):6–12, 1988. .
27. P. Pleinevaux and J.-D. Decotignie, Time critical communication networks: Field buses, *IEEE Network*, 2:55–63, 1988.
28. International Electrotechnical Commission, *IEC 61158–1, Digital Data Communications for Measurement and Control—Fieldbus for Use in Industrial Control Systems*, 2008. [Online]. www.iec.ch
29. International Electrotechnical Commission [IEC]. [Online]. www.iec.ch
30. International Organization for Standardization [ISO]. [Online]. www.iso.org
31. Instrumentation Society of America [ISA]. [Online]. www.isa.org
32. Comité Européen de Normalisation Electrotechnique [CENELEC]. [Online]. www.cenelec.org

33. European Committee for Standardization [CEN]. [Online]. www.cenorm.be
34. International Electrotechnical Commission, *IEC 61784–1, Digital Data Communications for Measurement and Control—Part 1: Profile Sets for Continuous and Discrete Manufacturing Relative to Fieldbus Use in Industrial Control Systems*, 2008. [Online]. www.iec.ch
35. International Electrotechnical Commission, IEC 61784–2: *Industrial Communication Networks—Profiles—Part 2: Additional Fieldbus Profiles for Real-Time Networks Based on ISO/IEC 8802–3*. [Online]. www.iec.ch
36. Schneider Automation, *Modbus Messaging on TCP/IP Implementation Guide*, May 2002, http://www.modbus.org
37. [Online]. www.modbus-ida.org
38. V. Shiffer, The CIP family of fieldbus protocols, *The Industrial Communication Technology Handbook*, CRC Press, Florida, 2005, pp. 14-1–14-65.
39. [Online]. www.odva.org
40. [Online]. www.controlnet.org
41. C. G. Jenkins, The anatomy of the P-NET fieldbus, *The Industrial Communication Technology Handbook*, CRC Press, Florida, 2005, pp. 15-1–15-25.
42. [Online]. http://www.yokogawa.com
43. [Online]. www.ethernet-powerlink.org
44. J. Feld, PROFINET—Scalable factory communication for all applications, *Proceedings of the 2004 IEEE International Workshop on Factory Communication Systems*, Vienna, Austria, September 22–24, 2004, pp. 33–38.
45. [Online]. www.profibus.org
46. S. C. Hibbard, P. Lutz, and R. M. Larsen, The SERCOS interface, *The Industrial Communication Technology Handbook*, CRC Press, Florida, 2005, pp. 38.1–38.30.
47. D. Jansen and H. Buttner, Real-time ethernet the EtherCAT solution, *Computing & Control Engineering Journal*, 15(1): 16–21, Febraury–March 2004.
48. [Online]. www.ethercat.org
49. M. Crevatin and T. P. von Hoff, HTTP digest authentication for embedded web servers, *Embedded System Handbook*, CRC Press, Florida, 2005, pp. 45.1–45.14.
50. J.-E. Frey, J. Endresen, A. Kreitz, and G. Scheible, Unplug but connected: Part 1. Redefining wireless, *ABB Review* 3, 2005.
51. J.-E. Frey, J. Endresen, A. Kreitz, and G. Scheible, Unplug but connected: Part 2. Wireless sensors and effectors in industrial control, ABB Review 3, 2005.
52. LAN/MAN Standards Committee of the IEEE Computer Society, *IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)*, October 2003.
53. HART Communication Foundation. [Online]. http://www.hartcomm2.org
54. ISA. [Online]. http://www.isa.org
55. Bluetooth Consortium, *Specification of the Bluetooth System*, 1999. [Online]. www.bluetooth.org
56. Bluetooth Special Interest Group, *Specification of the Bluetooth System*, Version 1.1, December 1999.
57. LAN/MAN Standards Committee of the IEEE Computer Society, *IEEE Standard for Information Technology—Telecommunications and Information Exchange between Systems—Local and Metropolitan Networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher Speed Physical Layer (PHY) Extension in the 2.4 Ghz Band*, 1999.
58. LAN/MAN Standards Committee of the IEEE Computer Society, *Information Technology—Telecommunications and Information Exchange between Systems—Local and Metropolitan Area*

Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999.
59. Institute of Electrical and Electronic Engineering Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band, June 2003, aNSI/IEEE Std 802.11.
60. D. Dzung, C. Apneseth, and J. Endresen, A wireless sensor/actuator communication system for real-time factory applications, *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation; ETFA 2005*, Catania, Italy, September 19–22, 2005.
61. D. Snoonian, Smart buildings, *IEEE Spectrum*, 40(8):18–23, 2003.
62. D. Loy, D. Dietrich, and H. Schweinzer, *Open Control Networks*, Kluwer, Boston, 2004.
63. S. T. Bushby, BACnet: A standard communication infrastructure for intelligent buildings, *Automation in Construction*, 6(5–6):529–540, 1997.
64. ENV 13154-2 (1998): *Data Communication for HVAC Applications—Field Net-Part 2: Protocols*.
65. EIA/CEA 776.5-1999, *CEBus-EIB Router Communications Protocol—The EIB Communications Protocol*.
66. EN 50090-X (1994–2004), *Home and Building Electronic Systems (HBES)*.
67. Konnex Association, Diegem, Belgium. *KNX Specifications, V. 1.1, 2004*.
68. [Online]. www.echelon.com
69. ANSI/EIA/CEA-709.1-A-1999, *Control Network Protocol Specification*.
70. EIA/CEA Std. 709.1, Rev. B, 2002, *Control Network Protocol Specification*.
71. ANSI/EIA/CEA 852–2002, *Tunneling Component Network Protocols over Internet Protocol Channels*.
72. [Online]. www.lonmark.org

2

Middleware Design and Implementation for Networked Embedded Systems

2.1 Introduction 2-1
Multiple Design Dimensions • Networked Embedded Systems Middleware • Example Application: Ping Node Scheduling for Active Damage Detection • Engineering Life Cycle • Middleware Design and Implementation Challenges

2.2 Middleware Solution Space 2-5
Problem: We Get More and/or Less Than We Need • Solution: Use a Bottom-Up Composition Approach to Get "Only" the Features That We Need • Problem: No Explicit Support for Temporal Coordination in the Simulation Environment • Solution: Virtual Clock Integrated with Real-Time Dispatching and Distribution Features

2.3 ORB Middleware for Networked Embedded Systems: A Case Study 2-8
Message Formats • Object Adapter • Message Flow Architecture • Time-Triggered Dispatching • Priority Propagation • Simulation Support

2.4 Design Recommendations and Trade-Offs 2-13
Use the Application to Guide Data-Types to Be Supported • Minimize Generality of the Messaging Protocol • Simplify Object Life-Cycle Management • Simplify Operation Lookup and Dispatch • Pay Attention to Safety and Liveness Requirements • Design to Support the Entire Engineering Life Cycle

2.5 Related Work 2-15

2.6 Concluding Remarks 2-15

Acknowledgments 2-15

References 2-16

Venkita Subramonian
AT&T Labs Research

Christopher D. Gill
Washington University

2.1 Introduction

Networked embedded systems support a wide variety of applications, ranging from temperature monitoring to battlefield strategy planning [6]. Systems in this domain are characterized by the following properties:

1. Highly connected networks
2. Numerous memory-constrained endsystems
3. Stringent timeliness requirements
4. Adaptive online reconfiguration of computation and communication policies and mechanisms

Networked embedded systems challenge assumptions about resource availability and scale made by classical approaches to distributed computing, and thus represent an active research area with many open questions. For example, advances in microelectro mechanical systems (MEMS) hardware technology have made it possible to move software closer to physical sensors and actuators to make more intelligent use of their capabilities. To realize this possibility, however, new networked embedded systems technologies are needed. For example, hardware infrastructure for such systems may consist of a network of hundreds or even thousands of small microcontrollers, each closely associated with local sensors and actuators.

2.1.1 Multiple Design Dimensions

The following four dimensions drive the design choices for development of many networked embedded systems:

1. Temporal predictability
2. Distribution
3. Feature richness
4. Memory constraints

There is often a contra-variant relationship between some of these design forces. For example, the left side of Figure 2.1 illustrates that feature richness may suffer when footprint is reduced. Similarly, a real-time embedded system's temporal performance must be maintained even when more or fewer features are supported, as illustrated by the right side of Figure 2.1.

Significant research has gone into each of these individual design dimensions and has resulted in a wide range of products and technologies. Research on the Embedded Machine [18] and Kokyu [10] mainly addresses the real-time dimension. The CORBA Event service [15], Real-time Publish/Subscribe [49], and Distributable Threads [28] provide alternative programming models that support both one-to-many and one-to-one communication and hence address the distribution dimension. Small footprint middleware is the main focus of e*ORB [34] and UCI-Core [40]. TAO [21]

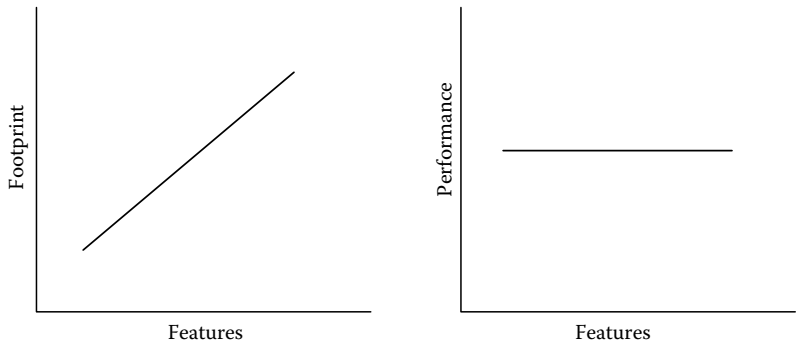


FIGURE 2.1 Features, footprint, and performance.

and ORBexpress RT [23] are general-purpose CORBA implementations that provide real-time and distribution features for a wide variety of application domains.

2.1.2 Networked Embedded Systems Middleware

General purpose middleware is increasingly taking the role that operating systems held three decades ago. Middleware based on standards such as CORBA [33], EJB [54], COM [39], and Java RMI [55] now caters to the requirements of a broad range of distributed applications such as banking transactions [4,24], online stock trading [5], and avionics mission computing [3]. Different kinds of general-purpose middleware have thus become key enabling technologies for a variety of distributed applications.

To meet the needs of diverse applications, general-purpose middleware solutions have tended to support a "breadth" of features. In large-scale applications, "layers" of middleware have been added to provide different kinds of services [3].

However, simply adding features breaks down for certain kinds of applications. In particular, features are rarely innocuous in applications with requirements for real-time performance or small memory footprint. Instead, every feature of an application and its supporting middleware is likely either to contribute to or detract from the application in those dimensions. Therefore, careful selection of features is crucial for memory constrained and/or real-time networked embedded systems.

As middleware is applied to a wider range of networked embedded systems, a fundamental tension between breadth of applicability and customization to the needs of each application becomes increasingly apparent. To resolve this tension, special-purpose middleware must be designed to address the following two design forces.

1. Middleware should provide common abstractions that can be reused across different applications in the same domain.
2. It should then be possible to make fine-grained modifications to tailor the middleware to the requirements of each specific application.

In the following section, we describe a motivating example application and the design constraints it imposes. In Section 2.1.4, we describe additional design constraints imposed by the engineering life cycle for this application.

2.1.3 Example Application: Ping Node Scheduling for Active Damage Detection

To illustrate how application domain constraints drive the design of special-purpose middleware, we now describe a next-generation aerospace application [9], in which a number of MEMS sensor/actuator nodes are mounted on a surface of a physical structure such as an aircraft wing. The physical structure may be damaged during operation, and the goal of this application is to detect such damage when it occurs. Vibration sensor/actuator nodes are arranged in a mesh with (wired or wireless) network connectivity to a fixed number of neighboring nodes. To detect possible damage, selected actuators called "ping nodes" generate vibrations which propagate across the surface of the physical structure. Sensors within a defined neighborhood can then detect possible damage near their locations by measuring the frequencies and strengths of these induced vibrations. The sensors convey their data to other nodes in the system, which aggregate data from multiple sensors, process the data to detect damage, and issue alerts or initiate mitigating actions accordingly.

Three restrictions on the system make the problem of damage detection difficult. First, the sensor/actuator nodes are resource-constrained. Second, two vibrations whose strengths are above a certain threshold at a given sensor location will interfere with each other. Third, sensor/actuator nodes may malfunction over time. These constraints, therefore, require that the actions of two

overlapping ping nodes be synchronized so that no interfering vibrations will be generated at a sensor location at any time.

This damage detection problem can be captured by a constraint model. Scheduling the activities of the ping nodes can be formulated as a distributed graph coloring problem. A color in the graph coloring problem corresponds to a specific time slot in which a ping node vibrates. Thus two adjacent nodes in the graph, each representing an actuator, cannot have the same color as the vibrations from these actuators would then interfere with each other. The number of colors is therefore the length (in distinct time slots) of a schedule. The problem is to find a minimal schedule such that the ping nodes do not interfere with one another, to minimize damage detection and response times. Distributed algorithms [57] have been shown to be effective for solving the distributed constraint satisfaction problem in such large scale and dynamic* networks.

2.1.4 Engineering Life Cycle

Large-scale networked embedded systems are often expensive and time-consuming to develop, deploy, and test. Allowing separate development and testing of the middleware and the target system hardware can reduce development costs and cycle times. However, this separation imposes additional design and implementation challenges for special-purpose middleware.

For example, to gauge performance of the distributed ping-scheduling algorithm in the actual system, physical, computational, and communication processes must be simulated for hundreds of nodes at once. For physical processes, tools such as Matlab or Simulink must be integrated within the simulation environment. Computation should be performed using the actual software that will be deployed in the target system. However, that software may be run on significantly different, and often fewer, actual endsystems in the simulation environment than in the target system. Similarly, communication in the simulation environment will often occur over conventional networks such as switched Ethernet which may not be representative of the target system's network.

The following issues must be addressed in the design and implementation of middleware that is suitable for both the simulation and target system environments:

- We need to use as much of the software that will be used in the target system as possible in the simulation environment. This helps us obtain relatively faithful metrics about the application and middleware that will be integrated with the target system.
- We need to allow arbitrary configurations for the simulation. The hardware and software configuration may be different for each machine used to run the simulation, and different kinds and numbers of target system nodes may be simulated on each machine.
- Simple time scaling will not work as it does not guarantee that the nodes are synchronized. First, it is not practical to require that all the computation and communication times are known *a priori*, as one function of the simulation may be to gauge those times. Moreover, even if we could scale the time to a "safe" upper bound, the wall-clock time it takes to run the simulation would likely be prohibitively large.
- Because of the heterogeneous configuration of the simulation environment, some simulated nodes might run faster than others, leading to causal inconsistencies in the simulation [29,31].
- Additional infrastructure is thus necessary to encapsulate the heterogeneity of different simulation environments and simulate real-time performance on top of general-purpose operating systems and networks, with simulation of physical processes in the loop.

* For example, with occasional reconfiguration due to sensor/actuator failures online.

2.1.5 Middleware Design and Implementation Challenges

To facilitate exchanges of information between nodes as part of the distributed algorithm, a middleware framework that provides common services like remote object method invocation is needed. Two key factors that motivate the development of ORB-style middleware for networked embedded systems are (1) remote communication and (2) location independence.

Remote communication: Even though a fixed physical topology may connect a group of sensor/actuator components, the "logical" grouping of these components may not strictly follow the "physical" grouping.

Location independence: The behavior of communicating components should be independent of their location to the extent possible. True location independence may not be achievable in all cases, e.g., due to timing constraints or explicit coupling to physical sensors or actuators. However, the implementation of object functionality should be able to be decoupled from the question of whether it accesses other objects remotely or locally where appropriate. The programming model provided to the object developer should thus provide a common programming abstraction for both remote and local access.

In summary, the key challenges we faced in the design and implementation of special-purpose middleware to address the application domain constraints described in Sections 2.1.3 and 2.1.4 are to

Reuse existing infrastructure: We want to avoid developing new middleware from scratch. Rather, we want to reuse pre-built infrastructure to the extent possible.

Provide real-time assurances: The performance of middleware itself must be predictable to allow application-level predictability.

Provide a robust DOC middleware: We chose the DOC communication paradigm as it offers direct communication among remote and local components, thus increasing location independence.

Reduce middleware footprint: The target for this middleware is memory-constrained embedded microcontroller nodes.

Support simulation environments: Simulations should be done with the same application software and middleware intended for deployment on the target. The middleware should also be able to deal with heterogeneous simulation testbeds, i.e., different processor speeds, memory resources, etc.

2.2 Middleware Solution Space

General-purpose CORBA implementations like TAO [48] offer generic CORBA implementations, whose feature sets are determined *a priori*. Furthermore, faithful implementation of the entire CORBA standard increases the number of features supported by ORBs and hence results in increased footprint for the application. In the case of memory-constrained networked embedded applications, this can become prohibitively expensive.

We instead want to get "only" the features that we need. The selection of features for our special-purpose middleware implementation was strictly driven by the unique requirements of the application domain. Two approaches to developing special-purpose middleware must then be considered:

- *Top-down*: subdividing existing general-purpose middleware frameworks, e.g., TAO [21]
- *Bottom-up*: composing special-purpose middleware from lower level infrastructure, e.g., ADAPTIVE Communication Environment (ACE) [22]

Both approaches seek to balance reuse of features with customization to application-specific requirements. The top-down approach is preferred when the number and kinds of features required

TABLE 2.1 Challenges and Potential Solutions

Challenge	Framework
Infrastructure reuse	ACE, TAO
Real-time assurances	Kokyu, TAO
Robust DOC middleware	TAO, e*ORB
Reduced middleware footprint	UIC-Core, e*ORB
Simulated real-time behavior	(TAO? Kokyu?)

are close to those offered by a general-purpose middleware implementation. In this case, provided policy and mechanism options can be adjusted in the general-purpose middleware to fit the requirements of the application. In general, this has been the approach used to create and refine features for real-time performance in TAO.

On the other hand, if the number or kinds of middleware features required differs significantly from those available in general-purpose middleware, as is the case with many networked embedded systems applications, then a bottom-up approach is preferable. This is based largely on the observation that in our experience lower-level infrastructure abstractions are less interdependent and thus more easily decoupled than higher-level ones. It is therefore easier to achieve highly customized solutions by composing middleware from primitive infrastructure elements [19,20] than trying to extract the appropriate subset directly from a general-purpose middleware implementation.

Modern software development relies heavily on reuse. Given a problem and a space of possible solutions, we try first to see whether the problem can be solved directly from an existing solution to a similar problem. Taking this view, we compared the challenges described in Section 2.1.5 to existing middleware solutions, as shown in Table 2.1.

TAO [21,48] and e*ORB [1,7] appeared to be the most suitable candidate solutions based on the requirements of our target application described in Section 2.1.3. TAO is a widely used standards-compliant ORB built using the ACE framework [22,42]. In addition to a predictable and optimized [36,56] ORB core [46], protocols [13,14], and dispatching [12,35] infrastructure, TAO offers a variety of higher level services [11,16]. e*ORB is a customized ORB that offers space-efficient implementation of a reduced set of features, with a corresponding reduction in footprint.

2.2.1 Problem: We Get More and/or Less Than We Need

Unfortunately, faithful implementation of the CORBA standard increases the number of features supported by TAO, e*ORB, and other similar CORBA implementations and hence results in increased footprint for the application. In the case of memory constrained applications, this becomes prohibitively expensive.

Although ACE reduces the complexity of the programming model for writing distributed object-oriented applications and middleware infrastructure, it does not "directly" address the challenges of real-time assurances, reduced footprint, or inter-operation with standards-based distribution middleware. Kokyu [10] is a low-level middleware framework built on ACE, for flexible multi-paradigm scheduling [8] and configurable dispatching of real-time operations. Thus, Kokyu can supplement the capabilities of another DOC middleware framework, but cannot replace it.

The UCI-Core approach supports different DOC middleware paradigms. It offers significant reuse of infrastructure, patterns, and techniques by generalizing features common to multiple DOC middleware paradigms and providing them within a minimal metaprogramming framework, thus also addressing the challenge of reducing middleware footprint. However, it is unsuited to meet other challenges described in Section 2.1.5, e.g., it does not directly support real-time assurances or simulation of real-time behavior.

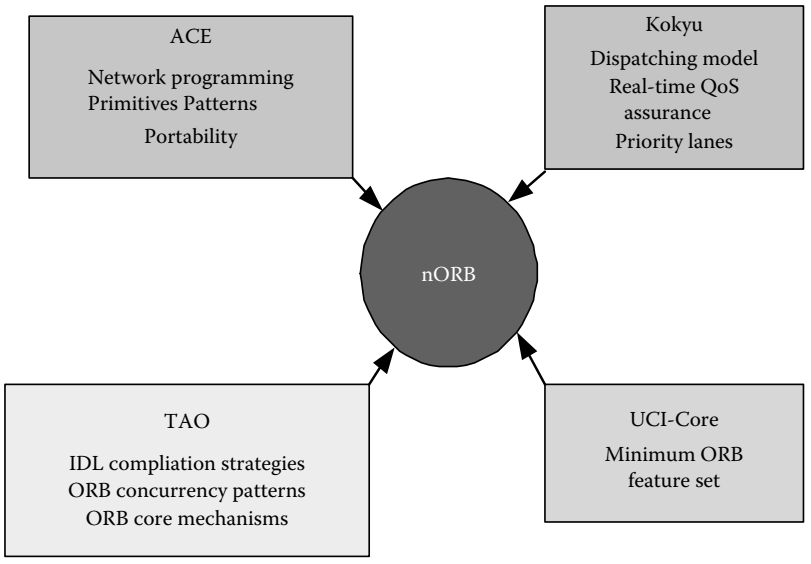


FIGURE 2.2 Reuse from existing frameworks.

2.2.2 Solution: Use a Bottom-Up Composition Approach to Get "Only" the Features That We Need

Figure 2.2 illustrates our approach. The selection of features for our special-purpose middleware implementation was strictly driven by the unique requirements of the application described in Section 2.1.3.

We initially considered a top-down approach, to avoid creating and maintaining an open-source code base separated from TAO. However, this approach proved infeasible due to several factors. First, the degree of implementation-level interdependence between features in TAO made it difficult to separate them. Second, the scarcity of mature tools to assist in identifying and decoupling needed vs. unneeded features made it unlikely we would be able to achieve such a top-down decomposition in a reasonable amount of time. Third, absent better tools it was also infeasible to validate that during refactoring we had correctly retained functional and real-time properties for the large body of TAO applications deployed outside our DOC middleware research consortium.

Therefore, we ultimately took a bottom-up compositional approach, which led to the development of nORB [9],* starting with the ACE framework and reusing as much as possible from it with transparent refactoring of some ACE classes to avoid unneeded features. By building on ACE, we reduced duplication between the TAO and nORB code bases, while achieving a tractable development process.

As in TAO, ACE components serve as primitive building blocks for nORB. Communication between nORB endsystems is performed according to the CORBA [33] model: the client side marshals the parameters of a remote call into a request message and sends it to a remote server, which then demarshals the request and calls the appropriate servant object; the reply is then marshaled into a reply message and sent back to the client, where it is demarshaled and its result returned to the caller. Although we did not retain strict compliance to the CORBA specification, wherever possible

* nORB is freely available as open-source software at http://deuce.doc.wustl.edu/nORB/

we reused concepts, interfaces, mechanisms, and formats from TAO's implementation of the CORBA standard.

2.2.3 Problem: No Explicit Support for Temporal Coordination in the Simulation Environment

As the last row of Table 2.1 suggests, none of the potential middleware solutions we found was able to support the kind of temporal coordination between simulated and actual infrastructure that is needed in the simulation environment described in Section 2.1.4.

Because TAO is open-source, it would be possible to integrate special-purpose mechanisms that intercept GIOP messages exchanged between simulated nodes, and perform accounting of simulation time whenever a message is sent or received. However, TAO does not explicitly address time-triggered dispatching of method invocations, which also must be subject to simulation time accounting.

Kokyu is designed for time- and event-triggered dispatching, but does not intercept messages exchanged through the ORB without transiting a dispatcher. Therefore, neither TAO nor Kokyu alone is able to provide the kind of temporal coordination needed in the simulation environment.

2.2.4 Solution: Virtual Clock Integrated with Real-Time Dispatching and Distribution Features

In the target system, both time-triggered local method invocations and remote method invocations must be dispatched according to real-time constraints. Those constraints and the corresponding temporal behavior of the application and middleware must be modeled and enforced effectively in the simulation environment as well.

To support both the target and simulation environments, we integrated a dispatcher based on the Kokyu model with distribution features based on TAO, in nORB. We then used the dispatcher as a single point of real-time enforcement, where both local upcalls and method invocation request and reply messages are ordered according to dispatching policies. Within that single point of control, we then integrated a virtual clock mechanism that is used only in the simulation environment, to enforce both causal consistency and real-time message and upcall ordering on the simulation's logical time-line.

2.3 ORB Middleware for Networked Embedded Systems: A Case Study

In this section, we describe the design and implementation of nORB, and the rationale behind our approach, to address the networked embedded system design and implementation challenges described in Section 2.1.5.

2.3.1 Message Formats

We support a limited subset of the messages supported by the CORBA specification, so that we do not incur unnecessary footprint, but at the same time support the minimum features required by the application. The following messages are supported by nORB: Request, Reply, Locate Request, and Locate Reply.

Figure 2.3 shows the formats of the messages supported by nORB. The format of the Request and Reply messages in nORB closely resembles that of the GIOP Request and Reply messages, respectively. We use the Common Data Representation [33] to encode the messages themselves. The nORB client builds a Request message and sends it to the nORB server, which sends a Reply back to the client.

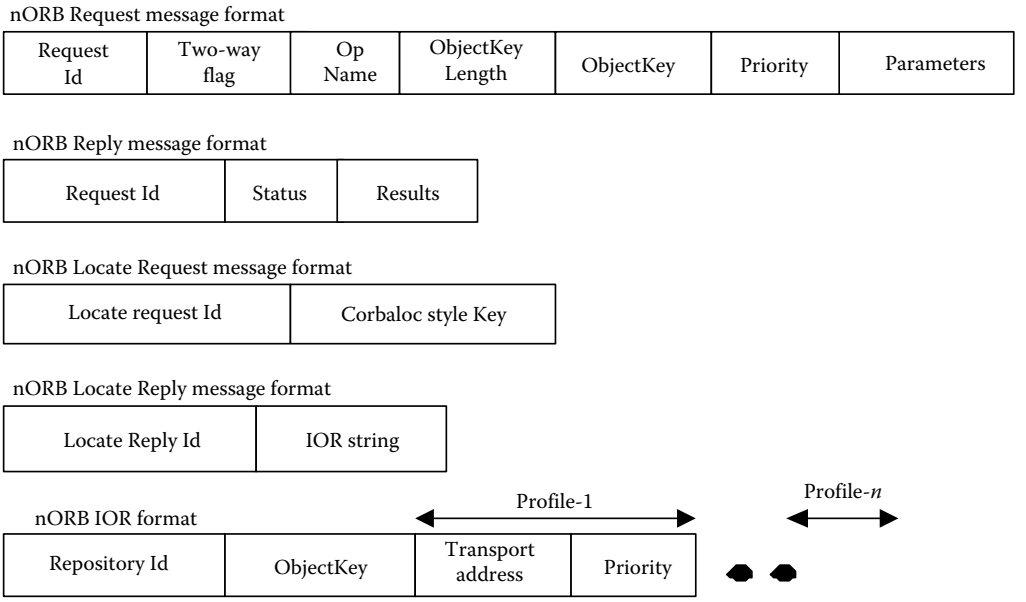


FIGURE 2.3 nORB IOR and message formats.

2.3.2 Object Adapter

In standard CORBA, each server-side ORB may provide multiple object adapters [38]. Servant objects register with an object adapter, which demultiplexes each client request to the appropriate servant. Each object adapter may be associated with a set of policies, e.g., for servant threading, retention, and lifespan [17]. In standard CORBA, multiple object adapters are supported by each ORB. This allows heterogeneous object policies to be implemented in a client-server environment, which is desirable in applications such as online banking, where each object on a server may be configured according to preferences of the server administrator, or even the end user.

In nORB, however, there is no assumption of multiple object adapters. Instead, a single object adapter per ORB is considered preferable for simplicity and footprint reduction. In nORB, the number of objects hosted on an embedded node is expected to be small, which reduces the need for multiple policies and thus for multiple object adapters. Even though the resulting object adapter does not conform to the Portable Object Adapter specification, a significant degree of footprint reduction is achieved because of the reduced object adapter functionality.

We have also simplified the process of object registration, to free developers from writing repetitive code as is seen in many CORBA programs. In the object adapter, we maintain a lookup table of object IDs and pointers to servant implementation objects. The lookup table is synchronized using a Readers/Writer lock. We have also consolidated object registration with other middleware initialization functions, by moving it from the object adapter interface to the ORB interface.

2.3.3 Message Flow Architecture

The message flow architecture in nORB uses strategies and patterns similar to those in TAO. We briefly mention the strategies we used and refer the interested reader to Ref. [43], which discusses these strategies in detail.

2.3.3.1 Reply Wait Strategy

When a client makes a remote two-way function call, the caller thread needs to block until it receives a reply back from the server. The two-way function call is made on the client stub, which then marshals the parameters into a Request and sends it to the server. The two-way function call semantics requires the caller thread to block until the reply comes back from the server. There are two different strategies to wait for the reply in TAO—"Wait on Connection" and "Wait on Reactor" [45]. nORB uses the Wait on Connection strategy to wait for the reply.

2.3.3.2 Upcall Dispatch Strategy

On the server side, there are different strategies to process an incoming request and send the reply back to the client. Two well-known strategies are "Direct Upcall" and "Queued Upcall" [45]. In the Direct Upcall strategy, the upcall is dispatched in the same thread as the I/O thread, which listens for incoming requests from the connection stream. The Queued Upcall strategy follows the Half Sync Half Async pattern [44,50]. In contrast with the direct upcall strategy, a network I/O thread is dedicated to receiving requests from clients. Once the request is received, it is encapsulated into a command object and then put into a queue. This queue is shared with another thread, in which the upcall dispatch is done. TAO uses the Direct Upcall strategy, whereas nORB uses the Queued Upcall strategy to address the simulation issues presented in Section 2.1.4, as is discussed in Section 2.3.6.

2.3.4 Time-Triggered Dispatching

In the Active Damage Detection application described in Section 2.1.3, control events must be triggered predictably at various rates. We developed a dispatcher for nORB to trigger these events predictably, based on the Kokyu [8] dispatching model. Kokyu abstracts combinations of fundamental real-time scheduling and dispatching mechanisms to enforce a variety of real-time policies, including well-known strategies such as Rate Monotonic Scheduling (RMS) [30], Earliest Deadline First [30], and Maximum Urgency First [51].

This dispatcher is used to trigger events on the client side and for dispatching upcalls on the server side. A "Dispatchable" interface is provided, which is to be implemented by an application object that needs to be time-triggered. The "handle_dispatch()" method is called on the Dispatchable object. In the ping-scheduling application, some application objects are both Dispatchable and make remote calls when triggered. Figure 2.4 shows the message flow from the client to the server when a timer expires on the client side, leading to a remote object method call on the server side.

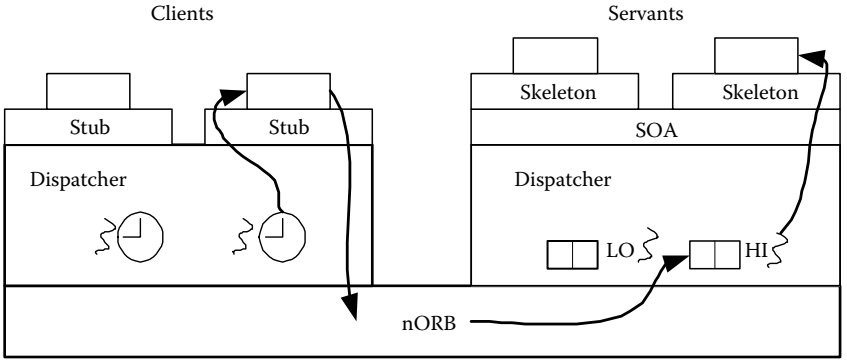


FIGURE 2.4 Kokyu dispatcher in nORB.

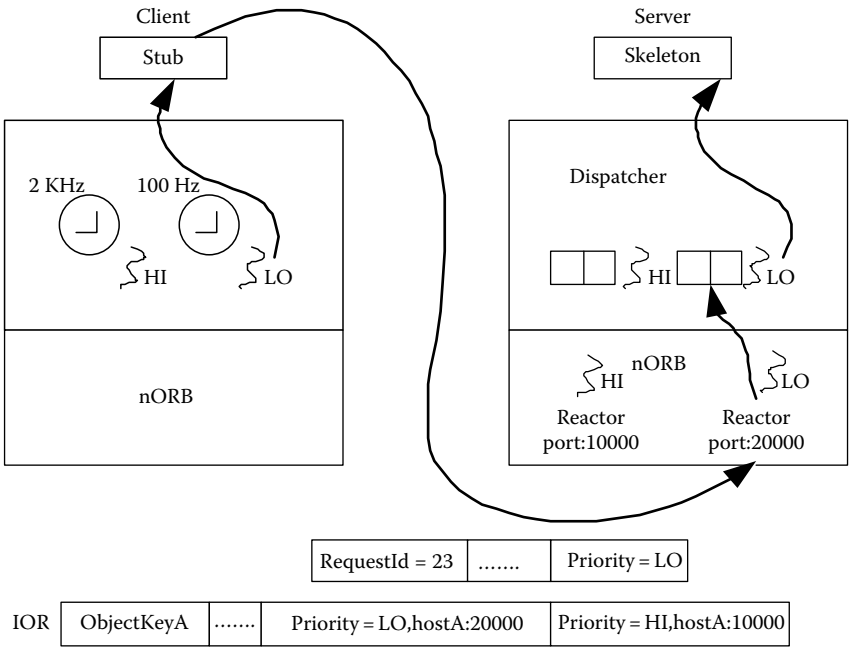


FIGURE 2.5 Priority propagation in nORB.

2.3.5 Priority Propagation

nORB implements real-time priority propagation in a way similar to TAO [46,47]. The client ORB uses the priority of the thread in which the remote invocation is made. nORB then looks for a matching priority from the set of profiles in the IOR and then makes a connection to the appropriate port. We use a cached connection strategy [43] to avoid the overhead of connection setup every time a remote invocation is made. To alleviate priority inversion, each connection endpoint on the server is associated with a thread/reactor pair, thus forming a "dispatching lane." The priority of the thread associated with each dispatching lane is set appropriately so that a request coming into a higher priority lane will be processed before a request coming into a lower priority lane. Figure 2.5 shows an example in which RMS [30] was used to assign priorities to the different rates on the client side.

The priority of the client thread making the request is propagated in the request. This priority is used on the server side to enqueue the request if necessary as explained in Section 2.3.6.

2.3.6 Simulation Support

Our solution to the engineering life-cycle issues described in Section 2.1.4 is to have nORB maintain a logical clock [2,29] at each node to support synchronization among nodes. We distinguish between "logical" times and priorities that are maintained by the simulation, and "actual" times and priorities that are maintained by the operating system. Logical time is divided into discrete frames. Each logical clock is incremented in discrete units by nORB after any incoming message is processed. If there are no incoming messages to be processed, then the logical time is incremented to the start of the next frame.

At the start of each frame, each node registered for a timed upcall in that frame is triggered, which results in the node waiting for sensor values from a physical simulation tool like Matlab or Simulink. While a node is waiting for the sensor values, we "freeze" the logical clock on that node, i.e., we

prevent the logical clock from advancing. As a result, no messages are processed on that node while it is waiting, and all incoming messages are queued. The queuing of incoming messages allows messages from the future to be delayed, and then processed after a slower node "catches up."

Any request or reply leaving a node carries the current logical clock value of that node. This value is used by the receiving node to sort the incoming messages based on their time of release and the logical clock of the receiving node. If the logical clock of the receiving node is later than that of the incoming message, then the message is stamped with the value of the logical clock at the receiving node.

Each "item" (dispatchable or request/reply message) carries its logical execution time, which is predefined for each item. When an item is ready and most eligible for execution, the clock thread dequeues it and checks whether it can complete its execution before the earliest more eligible item's release time. If it can complete its execution before another more eligible item must be processed, the clock thread enqueues the current item in the appropriate "lane" for "actual" execution on the processor. If not, the clock thread "simulates" the partial execution of the current item without "actually" executing it, by (1) storing the remaining logical execution time in the item itself and (2) enqueuing the updated item back into the clock thread's queue so that it can compete with other enqueued items for its next segment of execution eligibility.

A lane can be configured to run a single thread or a pool of worker threads. As described in Section 2.3.4, without clock simulation each lane thread is run at its own actual OS priority. In the simulation environment, time and eligibility are accounted for by the logical clock thread, so all the lane threads are run at the same actual OS priority. Each lane still maintains its logical priority in thread-specific storage [50], so that the logical priority can be sent with the request messages and used for eligibility ordering, as it will be in the target system.

Figure 2.6 shows an illustrative sequence of events that results from the addition of the logical clock thread to nORB, assuming for simplicity that all items run to completion rather than being preempted.

1. When the logical clock advances, the clock thread goes through the list of dispatchables to see whether any are ready to be triggered. A "ready" dispatchable is one whose next trigger time is less than or equal to the current logical clock and whose previous invocation has completed execution. In general, the clock thread determines the earliest time a message or dispatchable will be run, and marks all items with that time as being ready.
2. Any ready dispatchables are released to the clock thread's queues, according to their assigned logical priorities.
3. The clock thread selects the most eligible ready item (message or dispatchable) from among its priority queues. The clock thread then enqueues the selected item in the appropriate priority lane of the dispatcher, where it will compete with other messages and locally released dispatchables.
4. The corresponding lane thread in the dispatcher dispatches the enqueued item. The resulting upcall might in turn invoke a remote call to a servant object, which we describe in the following steps 5–8.
5. The logical priority of the dispatchable or the message is propagated to the server side. Currently, the application scheduler uses RMS to decide the logical priority of the dispatchable based on its simulated rate. Each lane thread stores its assigned logical priority in thread-specific storage [50]. The actual OS priorities of all the lane threads are kept the same under the clock simulation mechanism.
6. An incoming message is accepted by the server's reactor thread and is enqueued for temporal and eligibility ordering by the clock thread. Note that there is only one reactor thread, which runs at an actual priority level between the clock and the lane threads' actual

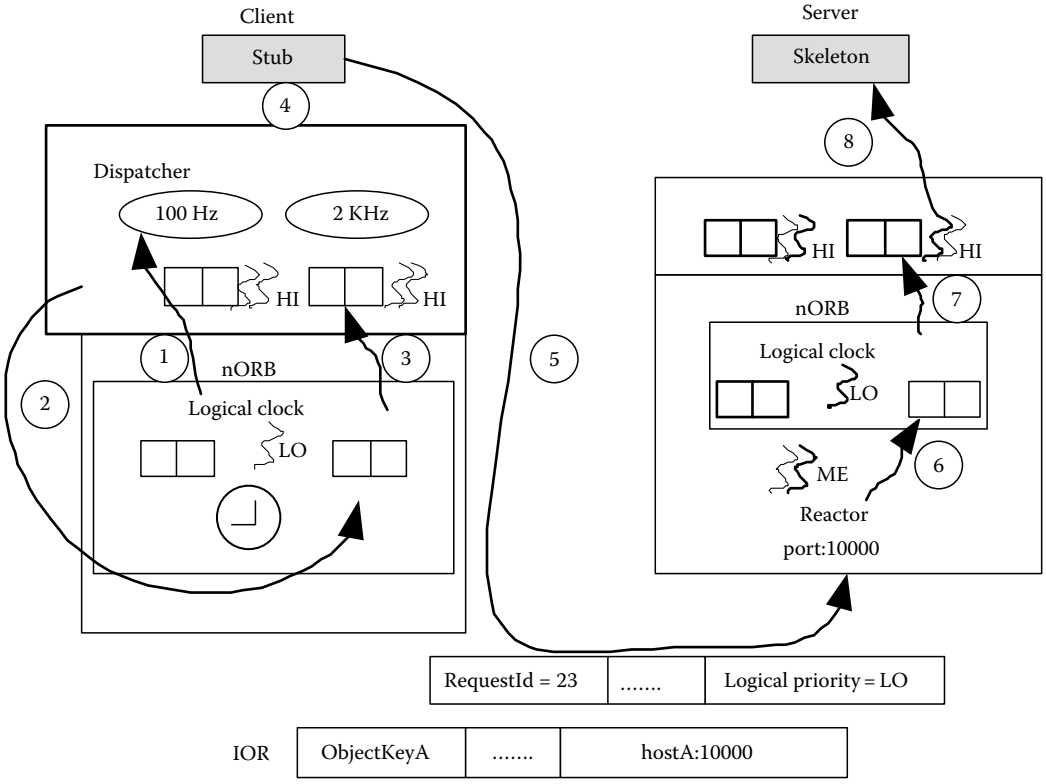


FIGURE 2.6 Processing with a logical clock.

priority levels. This is different from the previous approach discussed in Section 2.3.5 and is applied only when using the clock simulation. The lane threads are given the highest actual OS priority and the clock thread itself is assigned the lowest actual OS priority. This configuration of actual OS thread priorities reduces simulation times while still ensuring synchronization between nodes.

7. As on the client side, the clock thread then chooses the most eligible item from its queues and enqueues the item on the appropriate lane thread's queue.
8. The lane thread dispatches the enqueued item.

2.4 Design Recommendations and Trade-Offs

In this section, we present recommendations and trade-offs that we encountered in designing and developing the nORB special-purpose middleware to meet the challenges described in Section 2.1.5. While we focus specifically on our work on nORB, the same guidelines can be applied to produce middleware tailored to other networked embedded systems.

2.4.1 Use the Application to Guide Data-Types to Be Supported

We used the application domain to guide our choice of data types that the ORB middleware supports. In the damage detection application, e.g., sequences of simple structures are exchanged

between sensor/actuator nodes. nORB therefore supports only basic data types, structures, and sequences. This reduces the code base needed to support data types to those necessary for ping node scheduling. We supported marshaling in nORB to allow application deployment over heterogeneous networked embedded platforms, particularly for simulation environments. For other application domains with homogeneous platforms, support for marshaling could be removed entirely. To support other data types, e.g., CORBA Any, the middleware simply has to incorporate the code to marshal and demarshal those data types properly, trading off use of those types for an increase in footprint.

2.4.2 Minimize Generality of the Messaging Protocol

Previous work [37] has shown that optimizations can be achieved by the principle patterns of (1) relaxing system requirements and (2) avoiding unnecessary generality. Protocols in TAO like "GIOP-Lite" [37] are designed according to this principle. Similarly, as described in Section 2.3.1, we support a limited subset of the message types in the CORBA specification, so that we incur only the necessary footprint, while still providing all features required by our target application. By reducing the number of fields in the header, advanced features of a CORBA ORB such as Portable Interceptors are not supported. Providing this support would trade off with an increase in both ORB footprint and message sizes.

2.4.3 Simplify Object Life-Cycle Management

In a networked embedded system, the number of application objects hosted on each node is expected to be very small, which reduces the need for full-fledged life-cycle management. Servant objects are registered when the application starts, and live as long as the application, eliminating the need for more complicated dynamic life-cycle management. The trade-off is that, for large numbers of objects or where objects are dynamically created and destroyed, simplified object life-cycle management will not suffice. For such end-systems, more complex adapters are necessary, albeit at a cost of larger footprint.

2.4.4 Simplify Operation Lookup and Dispatch

When a remote operation request is received on a server, an ORB must search for the appropriate object in its Object Adapter and then perform a lookup of the appropriate method in the operations table. nORB uses linear search for that lookup because of the assumption that only a few methods on only a few objects on each node will receive remote calls. The linear search strategy reduces memory footprint while still maintaining real-time properties for small numbers of methods. The trade-off is that for large numbers of methods, real-time performance will suffer. This is because linear search will take $O(n)$ time to do a lookup, where n is the number of operations. Perfect hashing will take O(1) time, but this alternative would again entail increased footprint due to the code generated to support perfect hashing.

2.4.5 Pay Attention to Safety and Liveness Requirements

We described the different messaging and concurrency architecture choices in Section 2.3.3. With nested upcalls for remote method invocations, the "Wait on Connection" strategy could result in deadlocks [52]. The "Wait on Reactor" strategy, on the other hand, avoids deadlocks but introduces blocking factors which could hurt real-time performance [52].

2.4.6 Design to Support the Entire Engineering Life Cycle

Although many middleware solutions address the run-time behavior of networked embedded systems, few of them address earlier stages of the engineering life cycle. In particular, for networked embedded systems where simulation is used to gauge performance in the target system prior to system integration, additional special-purpose mechanisms may be needed. The virtual clock described in Section 2.3.6 is a good example of how such special-purpose mechanisms can be provided in middleware so that (1) application software is not modified for use in the simulation environment, and (2) the mechanism once developed can be reused for multiple applications.

2.5 Related Work

MicroQoSCORBA [32] focuses on footprint reduction through case-tool customization of middleware features. Ubiquitous CORBA projects [41] such as LegORB and the CORBA specialization of the Universally Interoperable Core (UIC) focus on a metaprogramming approach to DOC middleware. The UIC contains "meta-level" abstractions that different middleware paradigms, e.g., CORBA, must specialize, while ACE, TAO, and nORB are concrete "base-level" frameworks. e*ORB [34] is a commercial CORBA ORB developed for embedded systems, especially in the Telecommunications domain.

The time-triggered architecture (TTA) [27] is designed for fault-tolerant distributed real-time systems. Within the TTA, all system activities are initiated by the progression of a globally synchronized time-base. This stands in contrast to event-driven systems, in which system activity is triggered by events. The Time-triggered Message-triggered Object [25,26] architecture facilitates the design and development of real-time systems with syntactically simple but semantically powerful extensions of conventional object-oriented real-time approaches.

2.6 Concluding Remarks

We have described how meeting the constraints of networked embedded systems requires careful analysis of a representative application, "as an essential tool for the development of the special-purpose middleware itself." In addition, discovering "which" settings and features are best for an application requires careful design *a priori*. It is therefore important to adopt an iterative approach to middleware development that starts with specific application requirements and takes simulation and experimentation results into consideration.

By integrating both real-time middleware dispatching and a virtual clock mechanism used for simulation environments with distribution middleware features, we have shown how to develop special-purpose middleware solutions that address multiple stages of a networked embedded system's engineering life cycle. We also have empirically verified [53] that with nORB the footprint of a statically linked executable memory image for the ping node scheduling application was 30% of the footprint for the same application built with TAO, while still retaining real-time performance similar to TAO.

Acknowledgments

We gratefully acknowledge the support and guidance of the Boeing NEST OEP Principal Investigator Dr. Kirby Keller and Boeing Middleware Principal Investigator Dr. Doug Stuart. We also wish to thank Dr. Weixiong Zhang at Washington University in St. Louis for providing the initial algorithm implementation used in ping scheduling. This work was supported in part by the DARPA NEST (contract F33615-01-C-1898) and PCES (contract F33615-03-C-4111) programs.

References

1. S. Aslam-Mir. Experiences with real-time embedded CORBA in Telecom. In *OMG's First Workshop on Real-Time and Embedded Distributed Object Computing*, Falls Church, VA, July 2000. Object Management Group.
2. K. M. Chandy and L. Lamport. Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems*, 3(1), 63–75, Feb. 1985.
3. D. Corman. WSOA—weapon systems open architecture demonstration—Using emerging open system architecture standards to enable innovative techniques for time critical target (TCT) prosecution. In *Proceedings of the 20th IEEE/AIAA Digital Avionics Systems Conference (DASC)*, Daytona Beach, FL, Oct. 14–18, 2001.
4. L. R. David. Online banking and electronic bill presentment payment are cost effective.
5. X. D'efago, K. Mazouni, and A. Schiper. Highly available trading system: Experiments with CORBA, 1998.
6. D. Estrin, D. Culler, K. Pister, and G. Sukhatme. Connecting the physical world with pervasive networks. *IEEE Pervasive Computing*, 1(1), 59–69, Mar. 2002.
7. J. Garon. Meeting performance and QoS requirements with embedded CORBA. In *OMG's First Workshop on Embedded Object-Based Systems*, Santa Clara, CA, Jan. 2001. Object Management Group.
8. C. Gill, D. C. Schmidt, and R. Cytron. Multi-paradigm scheduling for distributed real-time embedded computing. *IEEE Proceedings, Special Issue on Modeling and Design of Embedded Software*, 91(1), 183–197, Jan. 2003.
9. C. Gill, V. Subramonian, J. Parsons, H.-M. Huang, S. Torri, D. Niehaus, and D. Stuart. ORB middleware evolution for networked embedded systems. In *Proceedings of the 8th International Workshop on Object Oriented Real-Time Dependable Systems (WORDS'03)*, Guadalajara, Mexico, Jan. 2003.
10. C. D. Gill, R. Cytron, and D. C. Schmidt. Middleware scheduling optimization techniques for distributed real-time and embedded systems. In *Proceedings of the 7th Workshop on Object-Oriented Real-Time Dependable Systems*, San Diego, CA, Jan. 2002. IEEE.
11. C. D. Gill, D. L. Levine, and D. C. Schmidt. The design and performance of a real-time CORBA scheduling service. *Real-Time Systems, The International Journal of Time-Critical Computing Systems, Special Issue on Real-Time Middleware*, 20(2), 117–154, Mar. 2001.
12. A. Gokhale and D. C. Schmidt. Evaluating the performance of demultiplexing strategies for real-time CORBA. In *Proceedings of GLOBECOM '97*, Phoenix, AZ, Nov. 1997. IEEE.
13. A. Gokhale and D. C. Schmidt. Principles for optimizing CORBA Internet inter-ORB protocol performance. In *Hawaiian International Conference on System Sciences*, Hawaii, Jan. 1998.
14. A. Gokhale and D. C. Schmidt. Optimizing a CORBA IIOP protocol engine for minimal footprint multimedia systems. *Journal on Selected Areas in Communications* special issue on Service Enabling Platforms for Networked Multimedia Systems, 17(9), 1673–1706, Sept. 1999.
15. T. H. Harrison, D. L. Levine, and D. C. Schmidt. The design and performance of a real-time CORBA event service. In *Proceedings of OOPSLA '97*, pp. 184–199, Atlanta, GA, Oct. 1997. ACM.
16. T. H. Harrison, D. L. Levine, and D. C. Schmidt. The design and performance of a real-time CORBA event service. *ACM SIGPLAN Notices*, 32(10), 184–200, Oct. 1997.
17. M. Henning and S. Vinoski. *Advanced CORBA Programming with C++*. Addison-Wesley, Reading, MA, 1999.
18. T. Henzinger, C. Kirsch, R. Majumdar, and S. Matic. Time safety checking for embedded programs. In *Proceedings of the Second International Workshop on Embedded Software* (*EMSOFT*), LNCS. Springer Verlag, New York, 2002.
19. F. Hunleth, R. Cytron, and C. Gill. Building customizable middleware using aspect oriented programming. In *The OOPSLA 2001 Workshop on Advanced Separation of Concerns in Object-Oriented Systems*, Tampa Bay, FL, Oct. 2001. ACM. www.cs.ubc.ca/~kdvolder/Workshops/OOPSLA2001/ASoC.html.

20. F. Hunleth and R. K. Cytron. Footprint and feature management using aspect-oriented programming techniques. In *Proceedings of the Joint Conference on Languages, Compilers and Tools for Embedded Systems*, pp. 38–45, ACM Press, Berlin, Germany, 2002.
21. Institute for Software Integrated Systems. The ACE ORB (TAO). www.dre.vanderbilt.edu/TAO/, Vanderbilt University.
22. Institute for Software Integrated Systems. The ADAPTIVE Communication Environment (ACE). www.dre.vanderbilt.edu/ACE/, Vanderbilt University.
23. O. Interface. ORBExpress. www.ois.com, 2002.
24. K. Kang, S. H. Son, and J. A. Stankovic. Star: Secure real-time transaction processing with timeliness guarantees, *23rd IEEE International Real-Time Systems Symposium* (RTSS 2002), Austin, TX, Dec. 3–5, 2002.
25. K. Kim. Object structures for real-time systems and simulators. *IEEE Computer Magazine*, (7), 62–70, Aug. 1997.
26. K. H. Kim. APIs enabling high-level real-time distributed object programming. *IEEE Computer*, 72–80, Jun. 2000.
27. H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, Norwell, MA, 1997.
28. Y. Krishnamurthy, C. Gill, D. C. Schmidt, I. Pyarali, L. M. Y. Zhang, and S. Torri. The design and implementation of real-time CORBA 2.0: Dynamic scheduling in TAO. In *Proceedings of the 10th Real-Time Technology and Application Symposium (RTAS '04)*, Toronto, CA, May 2004. IEEE.
29. L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 26(7), 558–565, Jul. 1978.
30. C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *The Journal of the ACM*, 20(1), 46–61, Jan. 1973.
31. N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1996.
32. A. D. McKinnon, K. E. Dorow, T. R. Damania, O. Haugan, W. E. Lawrence, D. E. Bakken, and J. C. Shovic. A configurable middleware framework with multiple quality of service properties for small embedded systems. In *2nd IEEE International Symposium on Network Computing and Applications*. IEEE Cambridge, MA, 197–204, Apr. 2003.
33. Object Management Group. *The Common Object Request Broker: Architecture and Specification*, 3.0.2 edition, Dec. 2002.
34. PrismTech. eORB. URL : http://www.prismtechnologies.com/English/Products/CORBA/eORB/.
35. I. Pyarali, C. O'Ryan, and D. C. Schmidt. A pattern language for efficient, predictable, scalable, and flexible dispatching mechanisms for distributed object computing middleware. In *Proceedings of the International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*, Newport Beach, CA, Mar. 2000. IEEE/IFIP.
36. I. Pyarali, C. O'Ryan, D. C. Schmidt, N. Wang, V. Kachroo, and A. Gokhale. Applying optimization patterns to the design of real-time ORBs. In *Proceedings of the 5th Conference on Object-Oriented Technologies and Systems*, pp. 145–159, San Diego, CA, May 1999. USENIX.
37. I. Pyarali, C. O'Ryan, D. C. Schmidt, N. Wang, V. Kachroo, and A. Gokhale. Using principle patterns to optimize real-time ORBs. *IEEE Concurrency Magazine*, 8(1), 2000.
38. I. Pyarali and D. C. Schmidt. An overview of the CORBA portable object adapter. *ACM StandardView*, 6(1), 16–25, Mar. 1998.
39. D. Rogerson. *Inside COM*. Microsoft Press, Redmond, WA, 30–43, 1997.
40. M. Roman. Ubicore: Universally Interoperable Core. www.ubi-core.com/Documentation/Universally_Interoperable_Core/universally_interoperable_core.html.
41. M. Román, R. H. Campbell, and F. Kon. Reflective middleware: From your desk to your hand. *IEEE Distributed Systems Online*, 2(5), 1–13 (online article), 2001.

42. D. C. Schmidt. ACE: An object-oriented framework for developing distributed applications. In *Proceedings of the 6th USENIX C++ Technical Conference*, Cambridge, MA, Apr. 1994. USENIX Association.
43. D. C. Schmidt and C. Cleeland. Applying a pattern language to develop extensible ORB middleware. In L. Rising, ed., *Design Patterns in Communications*. Cambridge University Press, New York, 393–438, 2000.
44. D. C. Schmidt and C. D. Cranor. Half-Sync/Half-Async: An architectural pattern for efficient and well-structured concurrent I/O. In *Proceedings of the 2nd Annual Conference on the Pattern Languages of Programs*, pp. 1–10, Monticello, IL, Sept. 1995.
45. D. C. Schmidt, D. L. Levine, and C. Cleeland. Architectures and patterns for developing high-performance, real-time ORB endsystems. In *Advances in Computers*, ed., Marvin V. Zelkowitz, Vol. 48, 1–118, Academic Press, London, 1999.
46. D. C. Schmidt, S. Mungee, S. Flores-Gaitan, and A. Gokhale. Alleviating priority inversion and non-determinism in real-time CORBA ORB core architectures. In *Proceedings of the 4th IEEE Real-Time Technology and Applications Symposium*, Denver, CO, Jun. 1998. IEEE.
47. D. C. Schmidt, S. Mungee, S. Flores-Gaitan, and A. Gokhale. Software architectures for reducing priority inversion and non-determinism in real-time object request brokers. *Journal of Real-Time Systems, Special Issue on Real-Time Computing in the Age of the Web and the Internet*, 21(2), 77–125, 2001.
48. D. C. Schmidt, B. Natarajan, A. Gokhale, N. Wang, and C. Gill. TAO: A pattern-oriented object request broker for distributed real-time and embedded systems. *IEEE Distributed Systems Online*, 3(2), 1–8 (online article), Feb. 2002.
49. D. C. Schmidt and C. O'Ryan. Patterns and performance of real-time publisher/subscriber architectures. *Journal of Systems and Software, Special Issue on Software Architecture—Engineering Quality Attributes*, 66, 223–239, 2002.
50. D. C. Schmidt, M. Stal, H. Rohnert, and F. Buschmann. *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects*, Vol. 2. Wiley & Sons, New York, 2000.
51. D. B. Stewart and P. K. Khosla. Real-time scheduling of sensor-based control systems. In W. Halang and K. Ramamritham, eds., *Real-Time Programming*. Pergamon Press, Tarrytown, New York, 144–150, 1992.
52. V. Subramonian and C. Gill. A generative programming framework for adaptive middleware. In *Hawaii International Conference on System Sciences, Software Technology Track, Adaptive and Evolvable Software Systems Minitrack, HICSS 2003*, Honolulu, HW, Jan. 2003. HICSS.
53. V. Subramonian, G. Xing, C. Gill, C. Lu, and R. Cytron. Middleware specialization for memory-constrained networked embedded systems. In *Proceedings of 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, Toronto, Canada, 2004.
54. Sun Microsystems, Inc. Enterprise JavaBeans Specification. java.sun.com/products/ejb/docs.html, Aug. 2001.
55. Sun Microsystems, Inc. *Java Remote Method Invocation Specification* (*RMI*), Oct. 1998.
56. N. Wang, D. C. Schmidt, and S. Vinoski. Collocation optimizations for CORBA. *C++ Report*, 11(10), 47–52, Nov./Dec. 1999.
57. W. Zhang, G. Wang, and L. Wittenburg. Distributed stochastic search for constraint satisfaction and optimization: Parallelism, phase transitions and performance. In *Proceedings of AAAI Workshop on Probabilistic Approaches in Search*, Edmonton, Canada, Jul. 28, 2002.

II

Wireless Sensor Networks

3 Introduction to Wireless Sensor Networks *Stefan Dulman and Paul J. M. Havinga* **3**-1
Third Era of Computing • What Are Wireless Sensor Networks? • Typical Scenarios and Applications • Design Challenges • Conclusions

4 Architectures for Wireless Sensor Networks *Stefan Dulman, S. Chatterjea, and Paul J. M. Havinga* **4**-1
Sensor Node Architecture • Wireless Sensor Network Architectures • Data-Centric Architecture • Distributed Data Extraction Techniques • Conclusion

5 Overview of Time Synchronization Issues in Sensor Networks *Weilian Su* ... **5**-1
Introduction • Design Challenges • Factors Influencing Time Synchronization • Basics of Time Synchronization • Time Synchronization Protocols for Sensor Networks • Conclusions

6 Resource-Aware Localization in Sensor Networks *Frank Reichenbach, Jan Blumenthal, and Dirk Timmermann* **6**-1
Introduction • Distance Estimation • Positioning Systems and Localization Algorithms • Conclusions

7 Power-Efficient Routing in Wireless Sensor Networks *Lucia Lo Bello and Emanuele Toscano* **7**-1
General Remarks on Routing in Wireless Sensor Networks • Overview of Energy-Saving Routing Protocols for WSNs • Data-Centric Power-Efficient Routing Protocols • Optimization-Based Power-Aware Routing Protocols • Cluster-Based Energy-Efficient Routing Protocols • Location-Based Energy-Aware Routing Protocols • Energy-Aware QoS-Enabled Routing Protocols • Topology Control Protocols for Energy-Efficient Routing • Summary and Open Issues

8 Energy-Efficient MAC Protocols for Wireless Sensor Networks *Lucia Lo Bello, Mario Collotta, and Emanuele Toscano* **8**-1
Design Issues for MAC Protocols for WSNs • Overview on Energy-Efficient MAC Protocols for WSNs • Mobility Support in WSNs • Multichannel Protocols for WSNs • Summary and Open Issues

9 Distributed Signal Processing in Sensor Networks *Omid S. Jahromi and Parham Aarabi* **9**-1
Introduction • Case Study: Spectrum Analysis Using Sensor Networks • Inverse and Ill-Posed Problems • Spectrum Estimation Using Generalized Projections • Distributed Algorithms for Calculating Generalized Projection • Conclusion

10 Sensor Network Security *Guenter Schaefer* **10**-1
Introduction • Denial of Service and Routing Security • Energy Efficient Confidentiality and Integrity • Authenticated Broadcast • Alternative Approaches to Key Management • Secure Data Aggregation • Summary

11 Wireless Sensor Networks Testing and Validation *Matthias Woehrle, Jan Beutel, and Lothar Thiele* **11**-1
Introduction • Wireless Sensor Network Validation • Sensor Network Testbeds • Integrated Testing Architecture • Test and Validation for Life on the Glacier—The PermaSense Case • Summary

12 Developing and Testing of Software for Wireless Sensor Networks *Jan Blumenthal, Frank Golatowski, Ralf Behnke, Steffen Prüter, and Dirk Timmermann* **12**-1
Introduction • Preliminaries • Software Architectures • Simulation, Emulation, and Test of Large-Scale Sensor Networks • Mastering Deployed Sensor Networks • Summary

3

Introduction to Wireless Sensor Networks

Stefan Dulman
University of Twente

Paul J. M. Havinga
University of Twente

3.1 Third Era of Computing 3-2
3.2 What Are Wireless Sensor Networks? 3-2
3.3 Typical Scenarios and Applications 3-3
3.4 Design Challenges 3-6
Locally Available Resources • Diversity and Dynamics • Needed Algorithms • Dependability
3.5 Conclusions 3-10
References 3-10

The start of the "Wireless Sensor Network" research field can be associated with the Smart Dust project [1]. Due to technological advances (and lots of imagination), building small-sized, energy-efficient reliable devices, capable of communicating with each other and organizing themselves in ad hoc networks became possible. These devices brought a new perspective to the world of computers as we know it: they could be embedded into the environment in such a way that the user is unaware of them. The need for reconfiguration and maintenance disappeared as the network organized itself to inform the users of the most relevant events detected or to assist them in their activity.

The first approach taken by the research community was to adapt the mature networking and data dissemination algorithms coming from the already existing networks to the problem at hand. It became pretty soon obvious that this approach had limited chances of success in front of the severe limiting factors the new technology imposed (lack of energy, lack of bandwidth, lack of communication power, dynamics of environment, unreliability, etc.). A new field of research was thus born from the need of solving (some already known) problems under a new and very restrictive set of assumptions.

This chapter will give a brief overview of the wireless sensor network area, by introducing the building concepts to the reader. Then, a number of applications as well as possible typical scenarios will be presented to better understand the field of application of this new emerging technology. Up to this moment, several main areas of applications have been identified. New areas of applications are still to be discovered as the research and products grow more mature.

Wireless sensor networks bring lots of challenges and often contradictory demands from the design point of view. The last part of the chapter will be dedicated to highlighting the main directions of research involved in this field. It will serve as a brief introduction to the problems to be described in the following chapters of the book.

3.1 Third Era of Computing

Things are changing continuously in the world of computers. Everything started with the mainframe era: some 30 years ago, these huge devices were widely deployed, for example, within universities. Lots of users made use of a single mainframe computer which they had to share among themselves. The computation power came together with a high cost and a huge machine requiring a lot of maintenance.

Technology advanced as it was predicted by Moore's law and we stepped into the second era of computers. It is a period that is still present today, but which is slowly approaching its final part. It is the era of the personal computers, cheaper and smaller, and increasingly affordable. Quite often, the average user has access to and makes use of more than one computer, these machines being present now in almost any home and work place.

But, in this familiar environment, things are starting to change and the third era of computing gains more and more terrain each day. Let us take a look at the main trends today. The technology advancements cause the personal computers to become smaller and smaller. The desktop computers tend to be replaced by laptops and other portable devices. What used to be a "regular" cell phone advanced to quite a sophisticated device (incorporating, for example, a digital camera and a personal digital assistant (PDA) having increasing communication capabilities, etc.).

The main factor that is influencing the new transition is the availability of wireless communication technology [1]. People are getting rapidly used to wireless communicating devices due to their independence from fixed machines. The success and availability of the Internet brought even more independence to the user: the data could now be available regardless of the physical location of its owner.

The advancements in technology did not stop here: the processors became small and cheap enough to be found now in almost any familiar device around us, starting with an everyday watch and ending with (almost) any home appliance we own. The new efforts nowadays are to make these devices "talk" to each other and organize themselves into ad hoc networks to accomplish their design goal as fast and reliably as possible.

This is, in fact, the third computer age envisioned two decades ago by Mark Weiser [2]. Several names such as ubiquitous computing, pervasive computing, ambient intelligence, invisible computing, disappearing computer, Internet of things, etc. were created to indicate different aspects of the new computing age (Mark Weiser himself defined it as "the calm technology, that recedes into the background of our lives").

The ubiquitous computing world brings a reversed view on the usage of computing power: instead of having lots of users gathered around the mainframe computer, now each user is using the services of several embedded networks. The user is in the middle of the whole system, surrounded by an invisible intelligent infrastructure. The original functionality of the objects and application is enhanced, and a continuous interaction is present in a large variety of areas of daily life.

So what is the next step? Technology will continue to advance and issues as miniaturization, number of features per device, battery lifetime and available connectivity everywhere, numbers of available sensors, etc. will be increasingly solved or at least improved. Computers (in various forms and shapes) start getting more and more accessories as sensors and "actuators." Getting used to the acting feature of these devices and designing applications specifically for a large number of devices to "do" things together will probably be a step forward in the development of embedded systems as we know them.

3.2 What Are Wireless Sensor Networks?

So what are wireless sensor networks and where is their place in this new environment that starts "growing" around us?

Wireless sensor networks is the generic name under which a broad range of devices hide. Basically, any collection of devices equipped with a processor, having sensing and communication capabilities and being able to organize themselves into a network created in an ad hoc manner falls into this category.

The addition of the wireless communication capabilities to sensors increased their functionality dramatically. Wireless sensor networks bring monitoring capabilities that changed forever the way in which data are collected from the ambient environment. Let us take, for example, the traditional monitoring approach of a remote location for a given phenomenon (such as recording the geological activity, monitoring the chemical or biological properties of a region, or even monitoring the weather at a certain place).

The old approach was the following: rather big and robust devices needed to be built. They should have contained besides the sensor pack itself, a big power supply, and local data storage capabilities. A team of scientists would have to travel together to the destination to be monitored, place these expensive devices at predefined positions, and calibrate all the sensors. Then, they would come back after a certain amount of time to collect the sensed data. If by misfortune some hardware would fail, then nothing could be done about it, as the information about the phenomenon itself would be lost.

The new approach is to construct inexpensive, small-sized, energy-efficient sensing devices. As hundreds or even thousands of these devices are to be deployed, the reliability constraints for them will be diminished. No local data storage is needed anymore as they will process locally and then transmit by wireless means the observed characteristic of the phenomenon to one or more access points connected to a computer network. Individual calibration of each sensor node is no longer needed as it can be performed by localized algorithms [3]. The deployment will also be easier, by randomly placing the nodes (e.g., simply throwing them from a plane) onto the monitored region.

Having this example in mind, we can give a general description of a sensor node. The name "sensor node" (or "mote") is used to describe a tiny device that has a short-range wireless communication capability, a small processor, and several sensors attached to it. It may be powered by batteries and its main function is to collect data from a phenomenon, collaborate with its neighbors, and forward its observations (preprocessed version of the data or even decisions) to the endpoint if requested. This is possible because its processor additionally contains the code that enables inter-node communication and setting-up, maintenance, and reconfiguration of the wireless network. When referring to wireless communication, we have in mind mainly radio communication (other means as ultrasound, visible or infrared light, etc. are also being used [4]). A "sensor network" is a network made up of large numbers of sensor nodes. By a large number, we understand at this moment hundreds or thousands of nodes but there are no exact limits for the upper bound of the number of sensors deployed.

Wireless sensor networks are one of the most important tools of the third era of computing. They are the simplest intelligent devices around, having as their main purpose monitoring the environment surrounding us and alerting us of the main events happening. Based on the observation reported by these instruments, humans and machines can make decisions and act on them.

3.3 Typical Scenarios and Applications

At this moment, a large variety of sensors exist. Sensors have been developed to monitor almost every aspect of the ambient world: lighting conditions, temperature, humidity, pressure, the presence or absence of various chemical or biological products, detection of presence and movement, etc. By networking large number of sensors and deploying them inside the phenomenon to be studied, we obtain a sensing tool way more powerful than a single sensor is able to do sensing at a superior level.

A first classification of wireless sensor networks can be made based on the complexity of the networks involved [5]:

- "*Intelligent*" *warehouse*—Each item to be monitored inside a warehouse is "tagged" with a small device; these tags are monitored by the fixed sensor nodes embedded into the walls and shelves. Based on the read data, knowledge of the spatial positioning of the sensors and time information, the sensor network offers information about the traffic of goods inside the building, creates automatic inventories, and even performs long-term correlations between the read data. The need of manual product scanning thus disappears. In this category, we can include the scenario of the modern supermarket, where the selected products of the customers are automatically identified at the exit of the supermarket. From the point of view of complexity, this scenario requires a minimum of complexity. The infrastructure sensor nodes are placed at fixed positions, in a more or less random manner. The deployment area is easily accessible and some infrastructures (e.g., power supplies and computers) already exist. At the same time, the nodes are operating in a "safe" controlled environment meaning that there are no major external factors that can influence or destroy them.
- *Environmental monitoring*—This is the widest area of applications envisioned up to now. A particular application in this category is disaster monitoring. The sensor nodes deployed in the affected areas can help humans estimate the effects of the disaster, build maps of the safe areas, and direct the human actions toward the affected regions. A large number of applications in this category address monitoring of the wildlife. This scenario has an increased complexity. The area of deployment is no longer accessible in an easy manner and no longer safe for the sensor nodes. There is hardly any infrastructure present, nodes have to be scattered around in a random manner and the network might contain moving nodes. Also a larger number of nodes need to be deployed.
- *Very large scale sensor networks applications*—The scenario of a large city where all the cars have integrated sensors. These sensor nodes communicate with each other collecting information about the traffic, routes, and special traffic conditions. On one hand, new information is available to the driver of each car. On the other hand, a global view of the whole picture can also be available. The two main constraints that characterize this scenario are the large number of nodes and their high mobility. The algorithms employed need to scale well and deal with a network with a continuously changing topology.

On the other hand, the authors of Ref. [6] present a classification of sensor networks based on their area of application. It takes into consideration only the military, environment, health, home, and other commercial areas, and can be extended with additional categories such as space exploration, chemical processing, and disaster relief.

- *Military applications*—Factors as rapid deployment, self-organization, and increased fault tolerance make wireless sensor networks a very good candidate for usage in the military field. They are suited to deployment in battlefield scenarios due to the large size of the network and the automatic self-reconfiguration at the moment of the destruction or unavailability of some sensor nodes [7]. Typical applications are monitoring of friendly forces, equipment, and ammunition; battlefield surveillance; reconnaissance of opposing forces and terrain, targeting, battle damage assessment; and nuclear, biological, and chemical attack detection and reconnaissance. A large number of projects have already been sponsored by The Defense Advanced Research Projects Agency.

- *Environmental applications*—Several aspects of the wildlife are being studied with the help of sensor networks. Existing applications include the following: monitoring the presence and the movement of birds, animals, and even insects; agricultural-related projects observing the conditions of crops and livestock; environmental monitoring of soil, water, and atmosphere contexts and pollution studies; etc. Other particular examples include forest fire monitoring, biocomplexity mapping of the environment, and flood detection. Ongoing projects at this moment include the monitoring of birds on Great Duck Island [8], the zebras in Kenya [9], the redwoods in California [10], or the water temperature of the Australian coral reef [11]. The number of these applications is continuously increasing as the first deployed sensor network shows the benefits of easy remote monitoring.
- *Healthcare applications*—An increasing interest is being shown to the elder population [12]. Sensor networks can help in several areas of the healthcare field. The monitoring can take place both at home and in hospitals. At home, patients can be under permanent monitoring and the sensor networks will trigger alerts whenever there is a change in the patient's state. Systems that can detect their movement behavior at home, detect any fall, or remind them to take their prescriptions are being studied. Also inside the hospitals sensor networks can be used to track the position of doctors and patients (their status or even errors in the medication), expensive hardware, etc. [13].
- *Home applications*—The home is the perfect application domain for the pervasive computing field. Imagine all the electronic appliances forming a network and cooperating together to fulfill the needs of the inhabitants [14]. They must identify each user correctly, remember their preferences and their habits, and, at the same time, monitor the entire house for unexpected events. The sensor networks have also an important role here, being the "eyes and the ears" that will trigger the actuator systems.
- *Other commercial applications*—This category includes all the other commercial applications envisioned or already built that do not fit in the previous categories. Basically, they range from simple systems as environmental monitoring within an office to more complex applications such as managing inventory control and vehicle tracking and detection. Other examples include incorporating sensors into toys and thus detecting the position of the children in "smart" kindergartens [15]; monitoring the material fatigue and the tensions inside the walls of a building, etc.

The number of research projects dedicated to wireless sensor networks has increased dramatically over the last years. A lot of effort has been invested in studying all possible aspects of wireless sensor networks. Please refer to Table 3.1 for a few examples. Also, a number of companies were created, initially start-ups from the universities performing research in the field. Some of the leading names in the field, valid at the date of writing this document, are listed in Table 3.2.

TABLE 3.1 List of Sensor Networks-Related Research Projects

Project Name	Research Area
CoSense [16]	Collaborative sensemaking (target recognition, condition monitoring)
EYES [17]	Self-organizing, energy-efficient sensor networks
PicoRadio [18]	Develop low-cost, energy-efficient transceivers
SensoNet [19]	Protocols for sensor networks
Smart Dust [1]	Cubic millimeter sensor nodes
TinyDB [20]	Query-processing system
WINS [21]	Distributed network access to sensors, controls, and processors
Cobis [22]	Business rules and integration with higher-level systems
Esense [23]	Integrating context information
Sensei [24]	Architecture for global integration of sensor and actuator networks

TABLE 3.2 Current Sensor Networks Companies List

Company Name	Headquarters Location	HTTP Address
Ambient Systems	The Netherlands	http://www.ambient-systems.net
CrossBow	San Jose, California	http://www.xbow.com
Dust Networks	Berkeley, California	http://dust-inc.com
Ember	Boston, Massachussets	http://www.ember.com
Millennial Net	Cambridge, Massachussets	http://www.millennialnet.com
SYS Technologies	San Diego, California	http://www.systechnologies.com
MeshNetics	Dresden, Germany	http://www.meshnetics.com
Sensicast	Needham, Massachussets	http://www.sensicast.com
Tendril Networks	Boulder, Colorado	http://www.tendrilnetworks.com
ArchRock	San Fracisco, California	http://www.archrock.com
EnOcean	Oberhaching, Germany	http://www.enocean.com

3.4 Design Challenges

When designing a wireless sensor network one faces on one hand the simplicity of the underlying hardware, and, on the other hand, the requirements that have to be met. To satisfy them, new strategies and new sets of protocols have to be developed [25–27]. In the following, we will address the main challenges that are present in the wireless sensor network field. The research directions involved and the open questions that still need to be answered will be presented as well.

To begin with, a high-level description of the current goals for the sensor networks can be synthesized as

- *Long life*—The sensor node should be able to "live" as long as possible using its own batteries. This constraint can be translated to a power consumption less than 100 μW. The condition arises from the assumption that the sensor nodes will be deployed in a harsh environment where maintenance is either impossible or has a prohibitively high price. It makes sense to maximize the battery lifetime (unless the sensor nodes use some form of energy scavenging). The targeted lifetime of a node powered by two AA batteries is a couple of years. This goal can be achieved only by applying a strict energy policy which will make use of power-saving modes and dynamic voltage-scaling techniques.
- *Small size*—The size of the device should be less than 1 mm^3. This constrain gave the sensor nodes the name of "smart dust," a name which gives a very intuitive idea about the final design. Recently, the processor and the radio were integrated in a chip having a size of approximately 1 mm^3. What is left is the antenna, the sensors themselves, and the battery. Advances are required in each of these three fields to be able to meet this design constraint.
- *Inexpensive*—The third high-level design constraint is about the price of these devices. To encourage large scale deployment, this technology must be really cheap, meaning that the targeted prices in the range of a couple of cents.

3.4.1 Locally Available Resources

Wireless sensor networks may consist of thousands of devices working together. Their small size comes also with the disadvantage of very limited resources availability (limited processing power, low-rate unreliable wireless communication, small memory footprint, and low energy). This raises the issue of designing a new set of protocols across the whole system.

Energy is of special importance and can by far be considered the most important design constraint. The sensor nodes are mainly powered by batteries. In most of the scenarios, due to the environment where they are deployed, it is impossible to have a human change their batteries. In some designs, energy scavenging techniques can also be employed. Still, the amount of energy available to the nodes can be considered limited and this is why the nodes have to employ energy-efficient algorithms to maximize their lifetime.

By taking a look at the characteristics of the sensor nodes, we notice that the energy is spent for three main functions: environment sensing, wireless communication, and local processing. Each of these three components has to be optimized to obtain minimum energy consumption. For the sensing of the environment component, the most energy efficient available sensors have to be used. From this point of view, we can regard this component as a function of a specific application and a given sensor technology.

The energy needed for transmitting data over the wireless channel dominates by far the energy consumption inside a sensor node. More than that, it was previously shown that it is more efficient to use a short-range multihop transmission scheme than sending data over large distances [6]. A new strategy characteristic to the sensor networks was developed based on a trade-off between the last two components and it is, in fact, one of the main characteristics of the sensor networks (see for example techniques developed in Refs. [28,29]). Instead of blindly routing packets through the network, the sensor nodes will act based on the content of the packet [30].

Let us suppose that a certain event took place. All nodes that sensed it characterized the event with some piece of data that needs to be sent to the interested nodes. Many similar data packets are created, or at least, some redundancy exists in the packets to be forwarded. To reduce the traffic, each node on the communication path should examine the contents of the packet it has to forward. Then it should aggregate all the data related to a particular event into one single packet, eliminating the redundant information. The reduction of traffic by using this mechanism is substantial. Another consequence of this mechanism is that the user will not receive any raw data, but only high-level characterizations of the events. This makes us think of the sensor network as a self-contained tool, a distributed network that collects and processes information.

From an algorithmic point of view, the local strategies employed by sensor nodes have as a global goal to extend the overall lifetime of the network. The notion of lifetime of the network usually hides one of the following interpretations: one can refer to it as the time passed since power on and a particular event such as the energy depletion of the first node or of 30% of the nodes, or even the moment when the network is split in several subnetworks. No matter which of these concepts will be used, the nodes will choose to participate in the collaborative protocols following a strategy that will maximize the overall network lifetime.

To be able to meet the goal of prolonged lifetime, each sensor node should

- Spend all the idle time in a deep power down mode, thus using an insignificant amount of energy.
- When active, employ scheduling schemes that take into consideration voltage and frequency scaling.

It is interesting to note, at the same time, the contradictory wireless industry trends and the requirements for the wireless sensor nodes. The industry focuses at the moment in acquiring more bits/s/Hz while the sensor nodes need more bits/euro/nJ. From the transmission range point of view, the sensor nodes need only limited transmission range to be able to use an optimal calculated energy consumption, while the industry is interested in delivering higher transmission ranges for the radios. Nevertheless, the radios designed nowadays tend to be as reliable as possible, while a wireless sensor network is based on the assumption that failures are regarded as a regular event.

Energy is not the only resource the sensor nodes have to worry about. The processing power and memory are also limited. Large local data storage cannot be employed, so strategies need to be developed to store the most important data in a distributed fashion and to report the important events to the outside world. A feature that helps dealing with these issues is the heterogeneity of the network. There might be several types of devices deployed. Resource-poor nodes should be able to ask more powerful nodes to perform complicated computations. At the same time, several nodes could associate themselves to perform the computations in a distributed fashion.

Bandwidth is also a constraint when dealing with sensor networks. The low-power communication devices used (most of the time radio transceivers) can only work in simplex mode. They offer low data rates due also to the fact that they are functioning in the free unlicensed bands where traffic is strictly regulated.

3.4.2 Diversity and Dynamics

As we already suggested, there may be several kinds of sensor nodes present inside a single sensor network. We could talk of heterogeneous sensor nodes from the point of view of hardware and software. From the point of view of hardware, it seems reasonable to assume that the number of a certain kind of devices will be in an inversely proportional relationship to the capabilities offered. We can assist to a tiered architecture design, where the resource-poor nodes will ask more powerful or specialized nodes to make more accurate measurements of a certain detected phenomenon, to perform resource-intensive operations or even to help in transmitting data at a higher distance.

Diversity can also refer to sensing several parameters and then combine them in a single decision, or in other words to perform data-fusion. We are talking about assembling together information from different kinds of sensors like: light, temperature, sound, smoke, etc. to detect, for example, that a fire has started.

Sensor nodes are to be deployed in the real world, most probably in harsh environments. This puts them in contact with an environment that is dynamic in many senses and has a big influence on the algorithms that the sensor nodes should execute. First, the nodes are deployed in a random fashion in the environment and in some cases, some of them can be mobile. Second, the nodes are subject to failures at random times and they are also allowed to change their transmission range to better suit their energy budget. This leads to the full picture of a network topology in a continuous change. The algorithms for the wireless sensor networks have as one of their characteristic the fact that they do not require a predefined well-known topology.

One more consequence of the real world deployment is that there are many factors influencing the sensors in contact with the phenomenon. Individual calibration of each sensor node will not be feasible and probably will not help much as the external conditions will be in a continuous change. The sensor network should calibrate itself as a whole, as a reply to the changes in the environment conditions. More than this, the network will be capable of self-configuration and -maintenance.

Another issue we need to talk about is the dynamic nature of the wireless communication medium. Wireless links between nodes can periodically appear or disappear due to the particular position of each node. Bidirectional links will coexist with unidirectional ones and this is a fact that the algorithms for wireless sensor networks need to consider.

3.4.3 Needed Algorithms

For a sensor network to work as a whole, some building blocks need to be developed and deployed in the vast majority of applications. Examples can include a localization mechanism, a time synchronization mechanism, and some sort of distributed signal processing. A simple justification can be that data hardly has any meaning if some position and time values are not available with it. Full, complex signal processing done separately at each node will not be feasible due to the resource constraints.

The self-localization of sensor nodes gained a lot of attention lately [31–34]. It came as a response to the fact that global positioning systems are not a solution due to high cost (in terms of money and resources) and it is not available or provides imprecise positioning information in special environments as indoors, etc. Information such as connectivity, distance estimation based on radio signal strength, sound intensity, time of flight, angle of arrival, etc. were used with success in determining the position of each node within degrees of accuracy using only localized computation.

The position information once obtained was not only used for characterizing the data, but also in designing the networking protocols, for example, leading to more efficient routing schemes based on the estimated position of the nodes [35].

The second important building block is the timing and synchronization block. Nodes are allowed to function in a sleep mode for long period, so periodic waking-up intervals need to be computed within a certain precision. However, the notion of local time and synchronization with the neighbors is needed for the communication protocols to perform well. Light-weight algorithms have been developed that allow fast synchronization between neighboring nodes using a limited number of messages. Loose synchronization is to be used, meaning that each pair of neighbor nodes are synchronized within a certain bound, while nodes situated multiple hops away might not be synchronized at all.

Global timing notion might not be needed at all in most of the applications. Due to the fact that many applications measure natural phenomenon such as temperature, where delays up to the order of seconds can be tolerated, the trade-off between latency and energy is preferred.

The last important block is the signal processing unit. A new class of algorithms has to be developed due to the distributed nature of wireless sensor networks. In their vast majority the signal processing algorithms are centralized algorithms that require a large computation power and the availability of all the data at the same time. Transmitting all the recorded data to all nodes is impossible in a dense network even from theoretical point of view, not to mention the needed energy for such an operation. The new distributed signal processing algorithms have to take into account the distributed nature of the network, the possible unavailability of data from certain regions due to failures and the time delays that might be involved.

3.4.4 Dependability

More than any other sort of computer network, the wireless sensor networks are subject to failures. Unavailability of services are considered "a feature" of these networks or "regular events" rather than some sporadic and highly improbable events. The probability for something going wrong is at least several orders of magnitude higher than in all the other computer networks.

All the algorithms have to employ some form of robustness in front of the failures that might affect them. On the other hand, this comes at the cost of energy, memory, and computation power, so it has to be kept at a minimum. An interesting issue is the one of the system architecture from the protocols point of view. In traditional computer networks, each protocol stack is designed for the worst case scenario. This scenario hardly ever happens simultaneously for all the layers and a combination of lower layer protocols could eliminate such a scenario. This leads to lot of redundancy in the sensor node, redundancy that costs important resources. The preferred approach is that of cross-layer designing and studying of the sensor node as a whole object rather than separate building blocks. This opens for a discussion about the topic of what is a right architecture for all the sensor networks and if a solution that fits all the scenarios makes sense at all.

Let us summarize the sources of errors the designer will be facing: nodes will stop functioning starting with even the (rough) deployment phase. The harsh environment will continuously degrade the performances of the nodes making them unavailable as the time passes. Then, the wireless communication medium will be an important factor to disturb the message communication and to affect the links and implicitly the network topology. Even with a perfect environment, collisions will occur due to the imprecise local time estimates and lack of synchronization. Nevertheless, the probabilistic scheduling policies and protocol implementations can be considered sources of errors.

Another issue that can be addressed as a dependability attribute is the security. The communication channel is opened and cannot be protected. This means that others are able to intercept and to disrupt the transmissions or even to transmit their own data. In addition to accessing private information, a third party could also act as an attacker that wants to disrupt the correct functionality of the network.

The security in a sensor network is a hard problem that still needs to be solved. Like almost any other protocol in this sort of networks, it has contradictory requirements: the schemes employed should be as light as possible while achieving the best results possible. The usual protection schemes require too much memory and too much computation power to be employed (the keys themselves are sometimes too big to fit into the limited available memory).

A real problem is how to control the sensor network itself. The sensor nodes will be too many to be individually accessible to a single user and might also be deployed in an inaccessible environment. By control, we understand issues as deployment and installation, configuration, calibration and tuning, maintenance, discovery, and reconfiguration. Debugging the code running in the network is completely infeasible, as at any point inside, the user has access only to the high-level-aggregated results. The only real debugging and testing can be done with simulators that prove to be invaluable resources in the design and analysis of the sensor networks.

3.5 Conclusions

This chapter was a brief introduction to the new field of wireless sensor networks. It provided a short overview of the main characteristics of this new set of tools that will soon enhance our perception capabilities regarding the ambient world.

The major challenges have been identified, some initial steps have been taken and early prototypes are already working. The following chapters of the book will focus on particular issues, giving more insight to the current state of the art in the field. The research in this area will certainly continue and there may come a time when sensor networks will be deployed all around us and will become regular instruments available to everyone.

References

1. Consortium: Smartdust. http://robotics.eecs.berkeley.edu/pister/SmartDust (2001).
2. Weiser, M.: The computer for the 21st century. *SIGMOBILE Mobile Computing and Communications Review* **3** (1999) 3–11.
3. Whitehouse, K., Culler, D.: Calibration as parameter estimation in sensor networks. In: *Proceedings of ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, Georgia (2002).
4. Want, R., Hopper, A., Falcao, V., Gibbons, J.: The active badge location system. *ACM Transactions on Information Systems* **10** (1992) 91–102.
5. Estrin, D., Govindan, R., Heidemann, J., Kumar, S.: Next century challenges: Scalable coordination in sensor networks. In: *MobiCom '99: Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking,* New York, ACM (1999) pp. 263–270.
6. Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: A survey. *Computer Networks (Elsevier) Journal* **38** (2002) 393–422.
7. Brooks, R.R., Ramanathan, P., Sayeed, A.M.: Distributed target classification and tracking in sensor networks. *Proceedings of the IEEE* **91** (2003) 1163–1171.
8. Polastre, J., Szewczyk, R., Mainwaring, A., Culler, D., Anderson, J. In: *Analysis of Wireless Sensor Networks for Habitat Monitoring.* Kluwer Academic Publishers, Norwell, MA (2004) 399–423.
9. Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh, L., Rubenstein, D.: Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In: *ASPLOS-X*, San Jose, California (2002).
10. Yang, S.: Redwoods go hightech: Researchers use wireless sensors to study California's state tree. *UCBerkeley News* (2003).

11. Bondarenko, O., Kininmonth, S., Kingsford, M.: Coral reef sensor network deployment for collecting real time 3-d temperature data with correlation to plankton assemblages. *International Conference on Sensor Technologies and Applications, SensorComm 2007*, Washington, DC, 14–20 Oct. 2007, pp. 204–209.
12. Society, I.C.S.: *Pervasive Computing*, 3.2—*Successful Aging* (2004)
13. Baldus, H., Klabunde, K., Muesch, G.: Reliable set-up of medical body-sensor networks. In: *Proceedings of the Wireless Sensor Networks, First European Workshop (EWSN 2004),* Berlin, Germany (2004).
14. Basten, T., Geilen, M., Groot, H., eds.: Omnia fieri possent. In: *Ambient Intelligence: Impact on Embedded System Design.* Kluwer Academic Publishers, Boston (2003) pp. 1–8.
15. Srivastava, M., Muntz, R., Potkonjak, M.: Smart kindergarten: Sensor-based wireless networks for smart developmental problem-solving environments (challenge paper). In: *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, Rome, Italy, ACM (2001) pp. 132–138.
16. Consortium: Cosense project. http://www2.parc.com/spl/projects/ecca (2006).
17. Consortium: Eyes european project. http://eyes.eu.org (2006).
18. Consortium: Picoradio. http://bwrc.eecs.berkeley.edu/Research/Pico_Radio (2006).
19. Consortium: Sensonet. http://users.ece.gatech.edu/w̃eilian/Sensor/index.html (2006).
20. Consortium: Tinydb. http://telegraph.cs.berkeley.edu/tinydb (2003).
21. Consortium: Wins. http://www.janet.ucla.edu/WINS (2000).
22. Consortium: Cobis. http://www.cobis-online.de (2006).
23. Consortium: Esense. http://www.ist-e-esense.org (2007).
24. Consortium: Sensei. http://sensei-project.eu (2008).
25. Estrin, D., Culler, D., Pister, K., Sukhatme, G.: Connecting the physical world with pervasive networks. *IEEE Pervasive Computing* **1** (2002) 59–69.
26. Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E.: A survey on sensor networks. *IEEE Communication Magazine* **40** (2002) 102–114.
27. Pottie, G.J., Kaiser, W.J.: Wireless integrated network sensors. *Communications of the ACM* **43** (2000) 51–58.
28. Chlamtac, I., Petrioli, C., Redi, J.: Energy-conserving access protocols for identification networks. *IEEE/ACM Transactions on Networking* **7** (1999) 51–59.
29. Schurgers, C., Raghunathan, V., Srivastava, M.B.: Power management for energy-aware communication systems. *ACM Transactions on Embedded Computing Systems* **2** (2003) 431–447.
30. Intanagonwiwat, C., Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., Silva, F.: Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking (TON)* **11** (2003) 2–16.
31. Bulusu, N., Heidemann, J., Estrin, D.: GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications* **7**(5) (2000) 28–34.
32. Doherty, L., Pister, K., Ghaoui, L.: Convex position estimation in wireless sensor networks. In: *Proceedings of IEEE INFOCOM*, Anchorage, AK (2001).
33. Langendoen, K., Reijers, N.: Distributed localization in wireless sensor networks: A quantitative comparison. *Computer Networks* (Elsevier) *43(4) (2003)*, special issue on Wireless Sensor Networks.
34. Evers, L., Dulman, S., Havinga, P.: A distributed precision based localization algorithm for ad-hoc networks. In: *Proceedings of Pervasive Computing (PERVASIVE 2004)* Linz/Vienna, Austria (2004).
35. Zorzi, M., Rao, R.: Geographic random forwarding (GERAF) for ad hoc and sensor networks: Energy and latency performance. *IEEE Transactions on Mobile Computing* **2**(4) (2003) 349–365.

4

Architectures for Wireless Sensor Networks

Stefan Dulman
University of Twente

S. Chatterjea
University of Twente

Paul J. M. Havinga
University of Twente

4.1 Sensor Node Architecture 4-2
4.2 Wireless Sensor Network Architectures 4-4
Protocol Stack Approach • EYES Project Approach
4.3 Data-Centric Architecture 4-11
Motivation • Architecture Description • Additional Features • Generalized Component Interfaces • Concluding Remarks
4.4 Distributed Data Extraction Techniques 4-18
Attributes of Queries • Essential Conceptual Building Blocks • In-Network Processing • Acquisitional Query Processing • Cross-Layer Optimization • Data-Centric Data/Query Dissemination • Concluding Remarks
4.5 Conclusion 4-31
References 4-32

The vision of ubiquitous computing requires the development of devices and technologies that can be pervasive without being intrusive. The basic component of such a smart environment is a small node with sensing and wireless communications capabilities, able to organize itself flexibly into a network for data collection and delivery. Building such a sensor network presents many significant challenges, especially at the architectural, protocol, and operating system level.

Although sensor nodes might be equipped with a power supply or energy scavenging means and an embedded processor that makes them autonomous and self-aware, their functionality and capabilities will be very limited. Therefore, collaboration between nodes is essential to deliver smart services in a ubiquitous setting. New algorithms for networking and distributed collaboration need to be developed. These algorithms will be key for building self-organizing and collaborative sensor networks that show emergent behavior and can operate in a challenging environment where nodes move, fail, and energy is a scarce resource.

The question that rises is how to organize the internal software and hardware components in a manner that will allow them to work properly and be able to adapt dynamically to new environments, requirements, and applications. At the same time the solution should be general enough to be suited for as many applications as possible. Architecture definition also includes, at the higher level, a global view of the whole network. Besides the deployment aspects (topology, placement of base stations, beacons, etc.) we are also interested in how the user can access the data in the network. The data extraction (data discovery, data query, data retrieval, etc.) is part of the architecture and is traditionally heavily linked to the networking layers. A truly flexible architecture will include data

extraction mechanisms independent from the networking layer and the specific topology of a given network.

This chapter addresses the architecture of sensor networks. We build our argumentation incrementally as follows: we introduce the layered design approach for sensor networks, then we move on to the concept of a data-centric architecture and its flexible interfacing system, and we end the description with a detailed discussion on data extraction mechanisms.

4.1 Sensor Node Architecture

Current existing technology already allows integration of functionality for information gathering, processing, and communication in a tight packaging or even in a single chip (e.g. Figure 4.1 presents the EYES sensor node [1]). The four basic blocks needed to construct a sensor node are (see Figure 4.2):

- *Sensor and actuating platform*: The sensors are the interfaces to the real world. They collect the necessary information and have to be monitored by the central processing unit. The platforms may be built in a modular way such that a variety of sensors can be used in the same network. A wide range of sensors (monitoring characteristics of the environment such as light, temperature, air pollution, pressure, etc.) is already employed in the various applications. Lately, the sensing unit of devices is increasingly extended to contain one or more actuators. This practice slowly moves our perception on sensor networks from being "our eyes" in the environment to being also a means of changing some aspects of the environment.
- *Processing unit*: It is the intelligence of the sensor node. This not only collects the information detected by the sensor but also communicates with the network the device is

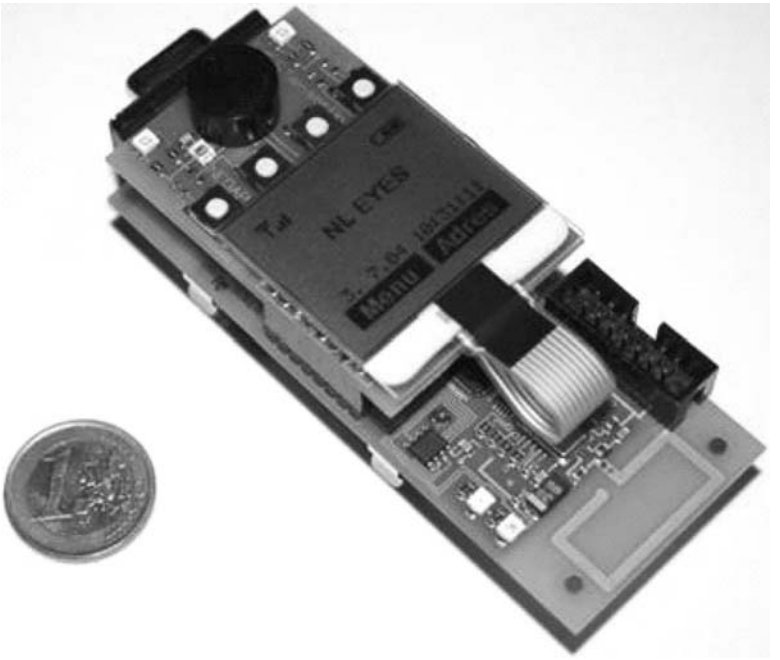


FIGURE 4.1 EYES sensor node. (From Consortium: EYES European project. http://eyes.eu.org, 2006.)

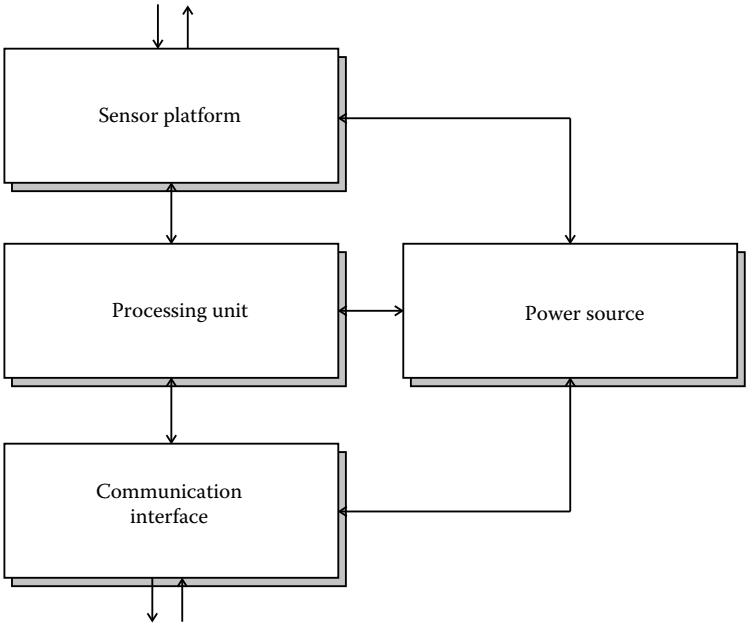


FIGURE 4.2 Sensor node components.

part of. The level of intelligence in the node strongly depends on the type of information that is gathered by its sensors and by the way in which the network operates. The sensed information is usually preprocessed to reduce the amount of data to be transmitted via the wireless interface. The processing unit also has to execute some networking protocols in order to forward the results of the sensing operation through the network to the requesting user.

- *Communication interface*: This is the link of each node to the sensor network itself. The focus relies on a wireless communication link, in particular on the radio communication, although visible or infrared light, ultrasound, etc. means of communications have already been used [2]. The used radio transceivers can usually function in simplex mode only, and can be completely turned off, in order to save energy. One of the important aspects of communication interfaces is that receiving information also consumes energy (in the case of the current off-the-shelf radio transceivers, the sending and receiving operations consume similar amounts of energy) so the assumption that a radio can have its receiver on always does not hold.
- *Power source*: Due to the application areas of the sensor networks, autonomy is an important issue. Sensor nodes are usually equipped with a power supply in the form of one or more batteries. Current practices focus on reducing the energy consumption by using low power hardware components and advanced networking and data management algorithms. The usage of energy scavenging techniques (converting solar energy, vibrations, temperature gradients, etc. into electrical power) makes it possible for the sensor nodes to be self-powered. No matter which form of power source is used, energy is still a scarce resource and a series of trade-offs will be employed during the design phase to minimize its usage.

Sensor networks are usually heterogeneous from the point of view of the types of nodes deployed. Moreover, whether or not any specific sensor node can be considered as being part of the network

only depends on the correct usage and participation in the sensor network suite of protocols and not on the node's specific way of implementing software or hardware. An intuitive description given in Ref. [3] envisions a sea of sensor nodes, some of them being mobile and some of them being static, occasionally containing tiny isles of relatively resource-rich devices. Some nodes in the system may perform autonomously (e.g., forming the backbone of the network by executing network and system services, controlling various information retrieval and dissemination functions, etc.), while others will have less functionality (e.g., just gathering data and relaying it to a more powerful node). Thus, from the sensor node architecture point of view we can distinguish between several kinds of sensor nodes. A simple yet sufficient in the majority of the cases approach would be to have two kinds of nodes: high-end sensor nodes (nodes that have plenty of resources or superior capabilities; the best candidate for such a node would probably be a fully equipped personal digital assistant (PDA) device or even a laptop) and low-end nodes (nodes that have only the basic functionality of the system and have very limited processing capabilities).

The architecture of a sensor node consists of two main components: defining the precise way in which functionality is needed and how to integrate it coherently in a sensor node. In other words, sensor node architecture means defining the exact way in which the selected hardware components connect to each other, how they communicate and how they interact with the central processing unit, etc.

A large variety of sensor node architectures have been built up to this moment. As a general design rule, all of them have targeted the following three objectives: energy efficiency, small size, and low cost. Energy efficiency is by far the most important design constraint because energy consumption depends on the lifetime of the sensor nodes. As the typical scenario of sensor networks deployment assumes that the power supplies of nodes will be limited and not rechargeable, a series of trade-offs need to be made to decrease the amount of consumed energy. Small size of the nodes leads to the ability of deploying lots of them to study a certain phenomenon. The ideal size is suggested by the name of one of the first research projects in the area: Smart Dust [4].

4.2 Wireless Sensor Network Architectures

A sensor network is a very powerful tool when compared to a single sensing device. It consists of a large number of nodes, equipped with a variety of sensors that are able to monitor different characteristics of a phenomenon. A dense network of such small devices, gives the user the opportunity to have a spatial view over the phenomenon and produces at the same time results based on a combination of various sorts of sensed data.

Each sensor node has two basic operation modes: initialization phase and operation phase. In order to guarantee good performance, the sensor network as a whole should function in a smooth way, with the majority of the nodes in the operation mode and only a subset of nodes in the initialization phase. The two modes of operation for the sensor nodes have the following characteristics:

- *Initialization mode*: A node can be considered in initialization mode if it tries to integrate itself in the network and is not performing its routine function. A node can be in initialization mode, e.g., at power on or when it detects a change in the environment and needs to configure itself. During initialization, the node can pass through different phases such as detecting its neighbors and the network topology, synchronizing with its neighbors, determining its own position, or even performing configuration operations on its own hardware and software. At a higher abstraction level, a node can be considered in initialization mode if it tries to determine which services are already present in the network, which services it needs to provide or can use.
- *Operation mode*: After the initialization phase the node enters a stable state, the regular operation state. It functions based on the conditions determined in the initialization

phase. The node can exit the operation mode and passes through an initialization mode if either the physical conditions around it or the conditions related to the network or to itself changed. The operation mode is characterized by small bursts of node activity (such as reading sensors values, performing computations, or participating in networking protocols) and periods spent in an energy-saving low power mode.

4.2.1 Protocol Stack Approach

A first approach to building a wireless sensor network (WSN) is to use a layered protocol stack as a starting point, as in the case of traditional computer network. The main difference between the two kinds of networks is that the blocks needed to build the sensor network usually span themselves over multiple layers while others depend on several protocol layers. This characteristic of sensor networks comes from the fact that they have to provide functionality that is not present in traditional networks. Figure 4.3 presents an example of mapping of the main blocks onto the traditional Open System Interconnection (OSI) protocol layers.

The authors of Ref. [5] propose an architecture based on the five OSI layers together with three management planes that go throughout the whole protocol stack (see Figure 4.4). A brief description of the layers included the physical layer addresses, mainly the hardware details of the wireless communication mechanism (the modulation type, the transmission and receiving techniques, etc). The data-link layer is concerned with the Media Access Control (MAC) protocol that manages communication over the noisy shared channel. Routing the data between the nodes is managed by the network layers, while the transport layer helps to maintain the data flow. Finally, the application layer contains (very often) only one single-user application.

In addition to the five network layers, three management planes have the following functionality: the power management plane coordinates the energy consumption inside the sensor node. It can, for example, based on the available amount of energy, allow the node to take part in certain distributed algorithms or to control the amount of traffic it wants to forward. The mobility management plane manages all the information regarding the physical neighbors and their movement patterns as well as its own moving pattern. The task management plane coordinates sensing in a certain region based on

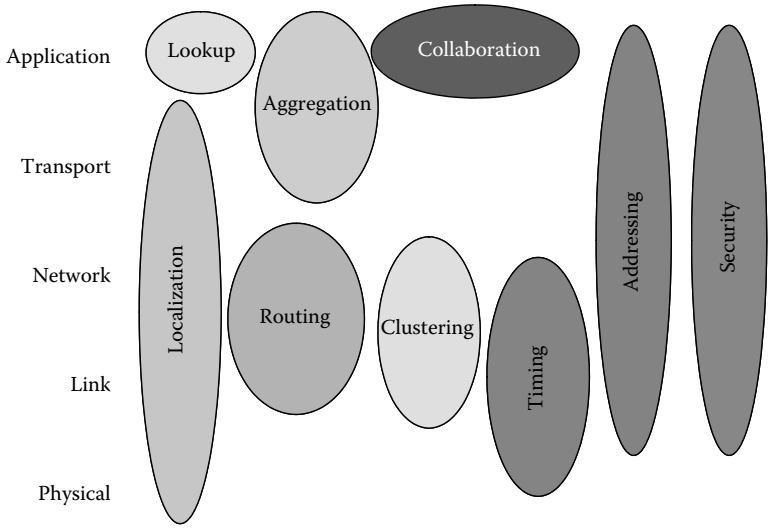


FIGURE 4.3 Relationship between building blocks/OSI layers.

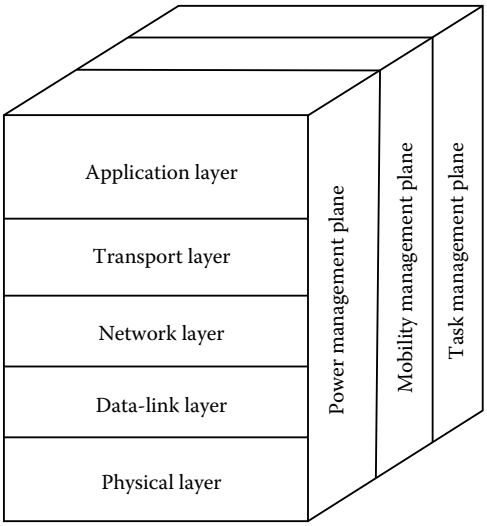


FIGURE 4.4 Protocol stack representation of the architecture. (From Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E., *IEEE Commun. Mag.*, 40, 102, 2002.)

the number of nodes and their placement (in very densely deployed sensor networks, energy might be saved by turning certain sensors off to reduce the amount of redundant information sensed).

In the following we give a description of the main building blocks needed to setup a sensor network. The description follows the OSI model. This should not imply that this is the right structure for these networks, but is offered merely as a reference point:

- *Physical layer*: It is responsible for the management of the wireless interface(s). For a given communication task, it defines a series of characteristics as operating frequency, modulation type, data coding, interface between hardware/software, etc.

  The large majority of already built sensor networks prototypes and most of the envisioned application scenarios assume the use of a radio transceiver as the means for communication. The unlicensed industrial, scientific, and medical (ISM) band is preferred because it is a free band designed for short-range devices using low-power radios and requiring low data-transmission rates. The modulation scheme used is another important parameter to decide upon. Complex modulation schemes are not preferred because they require important resources (in the form of energy, memory, and computation power). On the other hand, recent technological developments and efforts in standardization make the free 2.4 GHz communication band in combination with Zigbee technology [6] an attractive possibility for sensor networks.

  In the future, the advancements of the integrating circuits technology (e.g., ASIC, FPGA) will allow extended use of modulation techniques such as ultrawide band (UWB) or impulse radio (IR). Building general purpose sensor nodes using off-the-shelf components keeps (at least at the moment of writing this document) modulation techniques at a quite modest level.

  Based on the modulation type and on the hardware used, a specific data encoding scheme is chosen to assure both the synchronization required by the hardware component and a first level of error correction. At the same time, the data frame includes some carefully chosen preambles as well (series of initial bytes needed for the conditioning of the receiver circuitry and clock recovery).

It is worth mentioning that the minimum output power required to transmit a radio signal over a certain distance is directly proportional to the distance raised to a power between two and four (the coefficient depends on the type of the antenna used and its placement relative to the ground, indoor–outdoor deployment, etc.). In these conditions, it is more efficient to transmit a signal using a multihop network composed of short-range radios rather than using a (power consuming) long-range link [5].

The communication subsystem usually needs a hierarchy to create the abstraction for the other layers in the protocol stack (we are referring to the device hardware characteristics and the strict timing requirements). If a simple transceiver is used, some of these capabilities need to be provided by the main processing unit of the sensor node (this can require a substantial amount of resources for exact timing execution synchronization, cross-layer distribution of the received data, etc.). We notice an increasing complexity in the radio transceivers being used, with more and more functions moved to hardware and offered to the device via specialized interfaces (see, for example, the CC2431 transceiver [7] offering hardware capabilities as a link quality indicator, a security engine, a localization engine, etc.).

- *Data-link layer*: It is responsible for managing most of the communication tasks within one hop (both point-to-point and multicasting communication patterns). The main research issues here are the MAC protocols, the error control strategies and the power consumption control.

  The MAC protocols make the communication between several devices over a shared communication medium possible by coordinating the sending–receiving actions function of time and/or frequency. Several strategies have already been studied and implemented for the mobile telephony networks and for the mobile ad-hoc networks but unfortunately, none of them is directly applicable. Still, ideas can be borrowed from the existing standards and applications and new MAC protocols can be derived—this can be proved by the large number of new schemes that target specifically the WSNs.

  As the radio component is probably the main energy consumer in each sensor node, the MAC protocol must be very efficient. To achieve this, the protocol must, first of all, make use of the sleeping modes of the transceiver (e.g., turn the radio off) as much as possible because of the reduced energy consumption is these states. The most important problem comes from the scheduling of the sleep, receive, and transmit states. The transitions between these states also need to be taken into account as they consume energy and sometimes take large time intervals. Message collisions, overhearing, and idle listening are direct implications of the scheduling used inside the MAC protocol which, in addition, influences the bandwidth lost due to the control packet overheads.

  A second function of the data-link layer is to perform error control of the received data packets. The existent techniques include automatic repeat-request (ARQ) and forward error correction codes (FEC). The choice of a specific technique comes down to the trade-off between the energy consumed to transmit redundant information over the channel and the energy and high computation power needed at both the coder/decoder sides.

  Additional functions of the data-link layer are creating and maintaining a list of the neighbor nodes (all nodes situated within the direct transmission range of the node in discussion); extracting and advertising the source and destination as well as the data content of the overheard packets; supplying information related to the amount of energy spent on transmitting, receiving, coding, and decoding the packets, the amount of errors detected, the status of the channel, etc.

  Usually, several partially overlapping communication channels are available (the exact details depend on the transceiver being used). An interesting research topic is the usage

of multichannel MAC protocols. While this scheme traditionally falls into the category of frequency-based scheduling, a new interesting constraint is added to the problem: the usability of a given channel depends on the amount of power used in the adjacent communication bands. While the transmission power of the devices is usually kept constant, the distance between devices is not constant (and might also vary with time)—thus the topology of the network at a given moment is an important factor in deciding a particular channel allocation scheme [8].

- *Network layer*: This is responsible for routing of the packets inside the sensor network. It is one of the most studied topics in the area of WSN and a tremendous amount of literature is available on this topic. The main design constraint for this layer is, as in all the previous cases, the energy efficiency.

  The goal of WSNs is to deliver sensed data (or data aggregates) to the base stations requesting it. The concept of data-centric routing has been used to address this problem in an energy-efficient manner, minimizing the amount of traffic in the network. In data-centric routing, each node is assigned a specific task based on the interests of the base stations. In the second phase of the algorithm, the collected data is sent back to the requesting nodes. Interest dissemination can be done in two different ways, depending on the expected amount of traffic and level of events in the sensor network: the base stations can broadcast the interest to the whole network or the sensor nodes themselves can advertise their capabilities and the base stations will subscribe to that.

  Based on the previous considerations, the network layer needs to be optimized mainly for two operations: spreading the user queries, generated at one or more base stations, around the whole network and then retrieving the sensed data to the requesting node. Individual addressing of each sensor node is not important in the majority of the applications.

  Due to the high density of the sensor nodes, a lot of redundant information is available inside the sensor network. Retrieving all this information to a certain base station might easily exceed the available bandwidth, making the sensor network unusable. The solution to this problem is the data aggregation technique which requires each sensor node to inspect the content of the packets it has to route and aggregate the contained information, reducing the high redundancy of the multiple sensed data. This technique was proved to substantially reduce the overall traffic and make the sensor network behave as an instrument for analyzing data rather than just a transport infrastructure for raw data [9].
- *Transport layer*: This layer appears from the need to connect the WSN to an external network such as the Internet in order to disseminate its data readings to a larger community [10]. Usually the protocols needed for such interconnections require significant resources and they will not be present in all the sensor nodes. The envisioned scenario is to allow a small subset of nodes to behave as gateways between the sensor network and some external networks. These nodes will be equipped with superior resources and computation capabilities and will be able to run the needed protocols to interconnect the networks.

  As devices become more and more powerful and available bandwidth increases due to technological advances and improvements in the protocols, the transport layer attracts more attention. While probably not needed during the regular usage mode of a sensor network, there are cases (e.g., in the case of abnormal behavior detected by a device in the network) in which large amounts of data need to travel between the user and a specific node in the network. Examples also include the need of configuring or updating a software image on a given device.

- *Application layer*: It usually links the user's applications with the underlying layers in the protocol stack. Sensor networks are designed to fulfill one single application scenario for each particular case. The whole protocol stack is designed for a special application and the whole network is seen as an instrument. These make the application layer to be distributed along the whole protocol stack, and not appear explicitly. Still, for the sake of classification we can consider an explicit application layer that could have one of the following functionality [5]: sensor management protocol, task management and data advertisement protocol, and sensor query and data extraction protocol.

  Sensor networks reached the phase in which their integration with existing infrastructure is a topic of interest. From this perspective, the application layer can also be interpreted as a part of the middleware layer between existing infrastructure and the stream of live data provided by sensors. This interfacing is an interesting and difficult topic due to the mesh network characteristics. The design mentality under which it is developed is that the existing systems should not change at all—the sensor networks should adapt to what is already in place (which was designed usually at a time sensor networks were not present even as a concept).

4.2.2 EYES Project Approach

We bring now into discussion the architecture view proposed by the EYES European project [1]. This was the first European project dedicated to sensor networks and the architecture design was one of the major questions addressed. The approach taken in this project consists of only two key system abstraction layers: the sensor and networking layer and the distributed services layer (see Figure 4.5). Each layer provides services that may be spontaneously specified and reconfigured.

- *Sensor and networking layer*: It contains the sensor nodes (the physical sensor and wireless transmission modules) and the network protocols. Ad-hoc routing protocols allow messages to be forwarded through multiple sensor nodes taking into account the mobility of nodes and the dynamic change of topology. Communication protocols must be

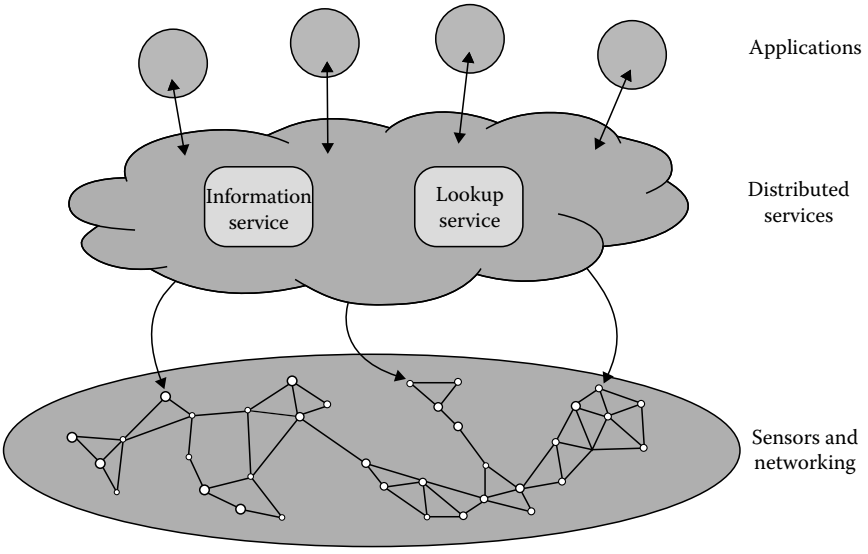


FIGURE 4.5 EYES project architecture description.

energy-efficient since sensor nodes have very limited energy supplies. To provide more efficient dissemination of data, some sensors may process data streams, and provide replication and caching.

- *Distributed services layer*: It contains distributed services for supporting mobile sensor applications. Distributed services coordinate with each other to perform decentralized services. These distributed servers may be replicated for higher availability, efficiency, and robustness. We have identified two major services. The lookup service supports mobility, instantiation, and reconfiguration. The information service deals with aspects of collecting data. This service allows vast quantities of data to be easily and reliably accessed, manipulated, disseminated, and used in a customized fashion by applications.

On top of this architecture applications can be built using the sensor network and distributed services. Communication in a sensor network is data-centric since the identity of the numerous sensor nodes is not important, only the sensed data together with time and location information counts. The three main functions of the nodes within a sensor network are directly related to this:

- *Data discovery*: Several classes of sensors will be equipped in the network. Specialized sensors can monitor climatic parameters (humidity, temperature, etc.), motion detection, vision sensors, and so on. A first step of data preprocessing can also be included in this task.
- *Data processing and aggregation*: This task is directly related to performing distributed computations on the sensed data and also aggregating several observations into a single one. The goal of this operation is the reduction of energy consumption. Data processing influences it by the fact that the transmission of one (raw sensed) data packet is equivalent to many thousands of computation cycles in the current architectures. Data aggregation keeps the overall traffic low by inspecting the contents of the routed packets, and in general, reducing the redundancy of the data in traffic by combining several similar packets into a single one.
- *Data dissemination*: This task includes the networking functionality comprising routing, multicasting, broadcasting, addressing, etc.

The existing network scenarios contain both static and mobile nodes. In some cases, the static nodes can be considered to form a back bone of the network and are more likely to be preferred in certain distributed protocols. Both mobile and static nodes will have to perform data dissemination, so the protocols should be designed to be invariant to node mobility. The particular hardware capabilities of each kind of sensor node determine how the previously described tasks should be mapped onto them (in principle all the nodes could provide all the previous functionality). During the initialization phase of the network, the functionality of every node is decided based on both the hardware configurations and the particular environmental conditions.

For a large sensor network to be able to function correctly, a tiered architecture is needed. This means that nodes will have to organize themselves into clusters based on certain conditions. The nodes in each cluster will elect a leader—the best fitted node to perform coordination inside the cluster (this can be, for example, the node with the highest amount of energy or the node having the most advanced hardware architecture or just a random node). The cluster leader is responsible for scheduling the node operations, managing the resources and the cluster structure, and maintaining communication with the other clusters.

We can talk about several types of clusters that can coexist in a single network:

- *Geographical clustering*: The basic mode of organizing the sensor network. The clusters are built based on the geographical proximity. Neighboring nodes (nodes that are in transmission range of each other) organize themselves into groups. This operation can

be handled in a completely distributed manner and it is a necessity for the networking protocols to work even when the network scales up.

- *Information clustering*: The sensor nodes can be grouped into information clusters based on the services they can provide. This clustering structure belongs to the distributed services layer and is built on top of the geographical clustering. Nodes using this clustering scheme need not be direct neighbors from the physical point of view.
- *Security clustering*: A even higher hierarchy appears if security is taken into consideration. Nodes can be grouped based on their trust levels or based on the actions they are allowed to perform or resources they are allowed to use in the network.

Besides offering increased capabilities to the sensor network, clustering is considered one of the principal building blocks for the sensor networks also from the point of view of energy consumption. The overhead given by the energy spent for creating and organizing the sensor network is easily recovered in the long term due to the reduced traffic it leads to.

4.3 Data-Centric Architecture

As we previously stated, the layered protocol stack description of the system architecture for a sensing node cannot cover all the aspects involved (such as cross-layer communication, dynamic update, etc.). The crossing planes though the protocol stack seems good as a solution, but in reality is almost difficult to implement. Establishing communication between the blocks in the protocol stack on a dynamic basis, being flexible with respect to the types of data being available at a certain moment inside the node and tolerating misbehaving blocks (correctly designed but producing unusable data due to effects of dynamics of the environment, mobility, out-of-date software, etc.) are some of the constraints affecting the design of a software architecture.

In this section we propose a new flexible architecture design. The architecture was designed having environmental dynamics in mind and is aimed at offering maximum flexibility while still adhering to the basic design concept of sensor networks: the devices should adapt at run time with the main goal of keeping the network alive, under a variety of conditions (the designer being probably unaware of most of them at design time).

4.3.1 Motivation

The sensor networks are dynamic from many points of view. Continuously changing behaviors can be noticed in several aspects of sensor networks, some of them being:

- *Sensing process*: The natural environment is dynamic by all means (the basic purpose of sensor networks is to detect, measure, and alert the user of the changing of its parameters). The sensor modules themselves can become less accurate, need calibration, or even break down.
- *Network topology*: One of the features of the sensor networks is their continuously changing topology. There are a lot of factors contributing to this, such as failures of nodes or the unreliable communication channel, mobility of the nodes, variations of the transmission ranges, clusters reconfiguration, addition/removal of sensor nodes, etc. Related to this aspect, the algorithms designed for sensor networks need to have two main characteristics: they need to be independent on the network topology and need to scale well with the network size.
- *Available services*: Mobility of nodes, failures, or availability of certain kinds of nodes might trigger reconfigurations inside the sensor network. The functionality of nodes may

depend on existing services at certain moments and when they are no longer available, the nodes either reconfigure or try to provide them themselves.

- *Network structure*: New kinds of nodes may be added to the network. Their different and increased capabilities will bring changes to the regular way in which the network functions. Software modules might be improved or completely new software functionality might be implemented and deployed in the sensor nodes.

Most WSN architectures currently use a fixed layered structure for the protocol stack in each node. This approach has certain disadvantages for WSNs. Some of them are:

- *Dynamic environment*: Sensor nodes address a dynamic environment where nodes have to reconfigure themselves to adapt to the changes. Since resources are very limited, reconfiguration is also needed in order to establish an efficient system (a totally new functionality might have to be used if energy levels drop under certain values). The network can adapt its functionality to a new situation, in order to lower the use of the scarce energy and memory resources, while maintaining the integrity of its operation.
- *Error control*: It normally resides in all protocol layers so that for all layers the worst case scenario is covered. For a WSN this redundancy might be too expensive. Adopting a central view on how error control is performed and cross-layer design reduces the resources spent for error control (and actually makes it feasible).
- *Power control*: It is traditionally done only at the physical layer, but since energy consumption in sensor nodes is a major design constraint, it is found in all layers (physical, data-link, network, transport, and application layer).
- *Protocol place in the sensor node architecture*: An issue arises when trying to place certain layers in the protocol stack. Examples may include: timing and synchronization, localization, and calibration. These protocols might shift their place in the protocol stack as soon as their transient phase is over. The data produced by some of these algorithms might make a different protocol stack more suited for the sensor node (e.g., a localization algorithm for static sensor networks might enable a better routing algorithm that uses information about the location of the routed data destination).
- *Protocol availability*: New protocols might become available after the network deployment or at certain moments, in specific conditions, some of the sensor nodes might use a different protocol stack that better suits their goal and the environment. The means of changing or updating at run time parts of the software on the nodes is an important needed feature.

4.3.2 Architecture Description

The system we are trying to model is an event-driven system, meaning that it reacts and processes the incoming events and afterward, in the absence of these stimuli, it spends its time in the power-down state (the software components running inside the sensor node are not allowed to perform blocking waiting).

We introduce a higher level of abstraction for the event class as data. Data may encapsulate the information provided by one or more events, have a unique name and contain additional information such as time information, identity of producer, etc. Data is the means used by the internal mechanisms of the architecture to exchange information components.

In the following we address any protocol or algorithm that can run inside a sensor node with the term entity (see Figure 4.6. An entity is a software component that will be triggered by the availability of one or more data types. While running, each entity is allowed to read available data types (but not wait for additional data types becoming available). As a result of the processing, each

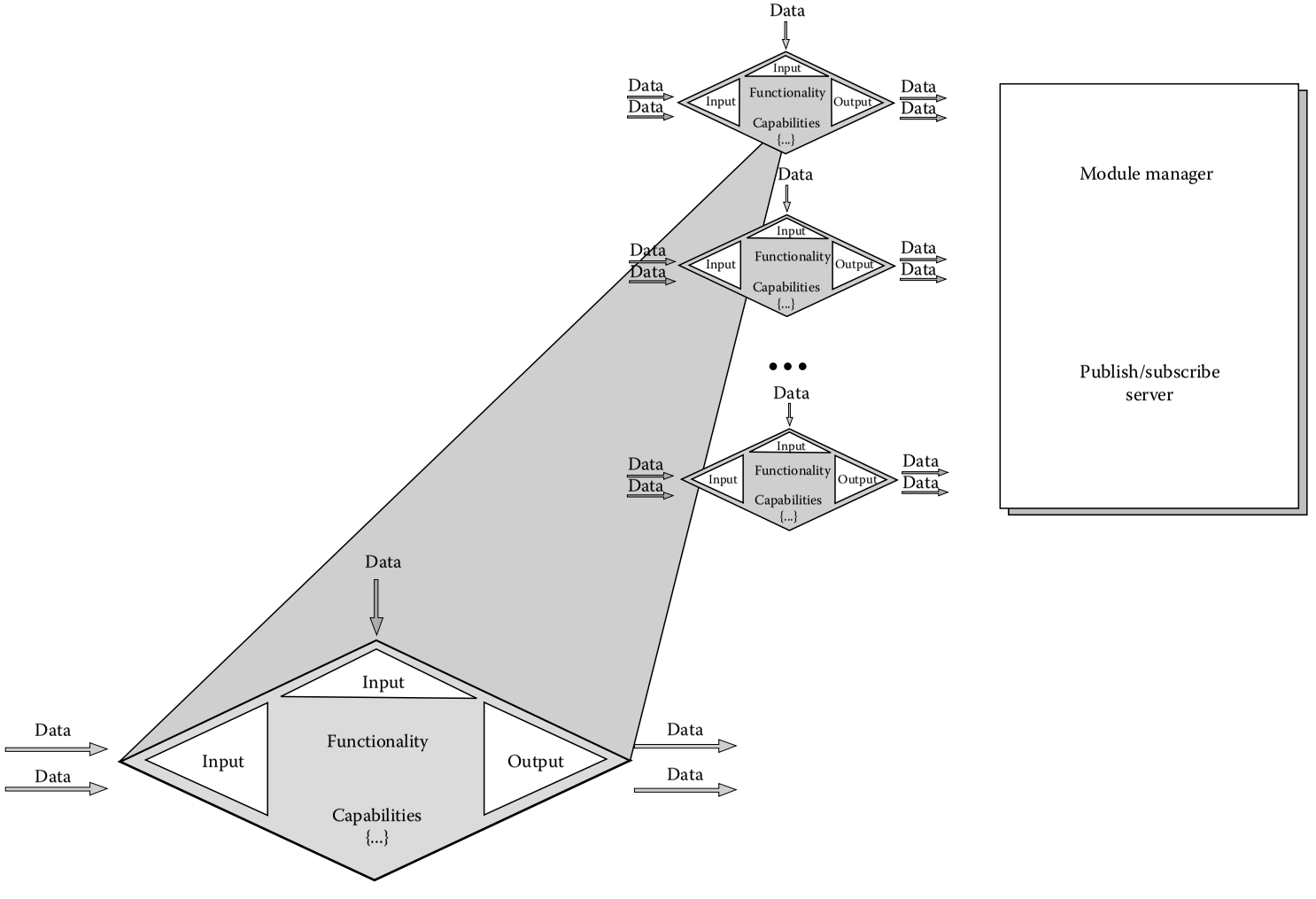


FIGURE 4.6 Entity description.

software component can produce one or more types of data (usually upon the completion of their execution).

An entity is also characterized by some functionality, meaning the sort of operation it can produce on the input data. Based on their functionality, the entities can be classified as being part of a certain protocol layer as in the previous description. For one given functionality, several entities might exist inside a sensor node; to discern among them, one should take into consideration their capabilities. By capability we understand high-level description containing the cost for a specific entity to perform its functionality (as energy, resources, time, etc.) and some characteristics indicating the estimated performance and quality of the algorithm.

In order for a set of components to work together, the way in which they have to be interconnected should be specified. Most of the architectures existent up to this moment in the WSN field, assume a fixed way in which these components can be connected, which is defined at compile time (except for the architectures that, for example, allow execution of agents). To change the protocol stack in such an architecture, the user should download the whole compiled code into the sensor node (via the wireless interface) and then make use of some boot loader code to replace the old running code in it. In the proposed architecture we are allowing this interconnection to be changed at run time, thus making on line update of the code possible, the selection of a more suited entity to perform some functionality based on the changes in the environment, etc. (in one word allowing the architecture to become dynamically reconfigurable).

To make this mechanism work, a new entity needs to be implemented—we call this the data manager. The data manager monitors the different kinds of data being available and coordinates the data flow inside the sensor node. At the same time it selects the most fitted entities to perform the work and it is even allowed to change the whole functionality of the sensor node based on the available entities and external environment (for a conceptual schematic see Figure 4.7.

The implementation of these concepts cannot make abstraction of the small amount of resources each sensor node has (as energy, memory, computation power, etc.). Going down from the abstraction level to the point where the device is actually working, a compulsory step is implementing the envisioned architecture in a particular operating system (in this case, due to the size and functionality provided, maybe a better term is system software). A large range of operating systems exist for embedded systems in general ([11,12] being two of the several tens of names). Scaled down versions with simple schedulers and limited functionality have been developed especially for WSN [13].

Usually, the issues of system architecture and operating system are treated separately, both of them trying to be as general as possible and to cover all the possible application cases. A simplistic view of a running operating system is a scheduler that manages the available resources and coordinates the execution of a set of tasks. This operation is centralized from the point of view of the scheduler that is allowed to take all the decisions. Our architecture can also be regarded as a centralized system, with the data manager coordinating the data flow of the other entities. To obtain the smallest overhead possible there should be a correlation between the function of the central nucleus from our architecture and the function of the scheduler from the operating system. This is why we propose a close relationship between the two concepts by extending the functionality of the scheduler with the functionality of the data manager. The main challenges that arise are keeping the size of the code low and the context-switching time.

4.3.3 Additional Features

As we mentioned earlier, the general concept of data is used rather than the event one. For the decision based on data to work, there are some additional requirements to be met.

First of all, all the modules need to declare the name of the data that triggers their action, the name of the data they need to read during their action (this can generically incorporate all the

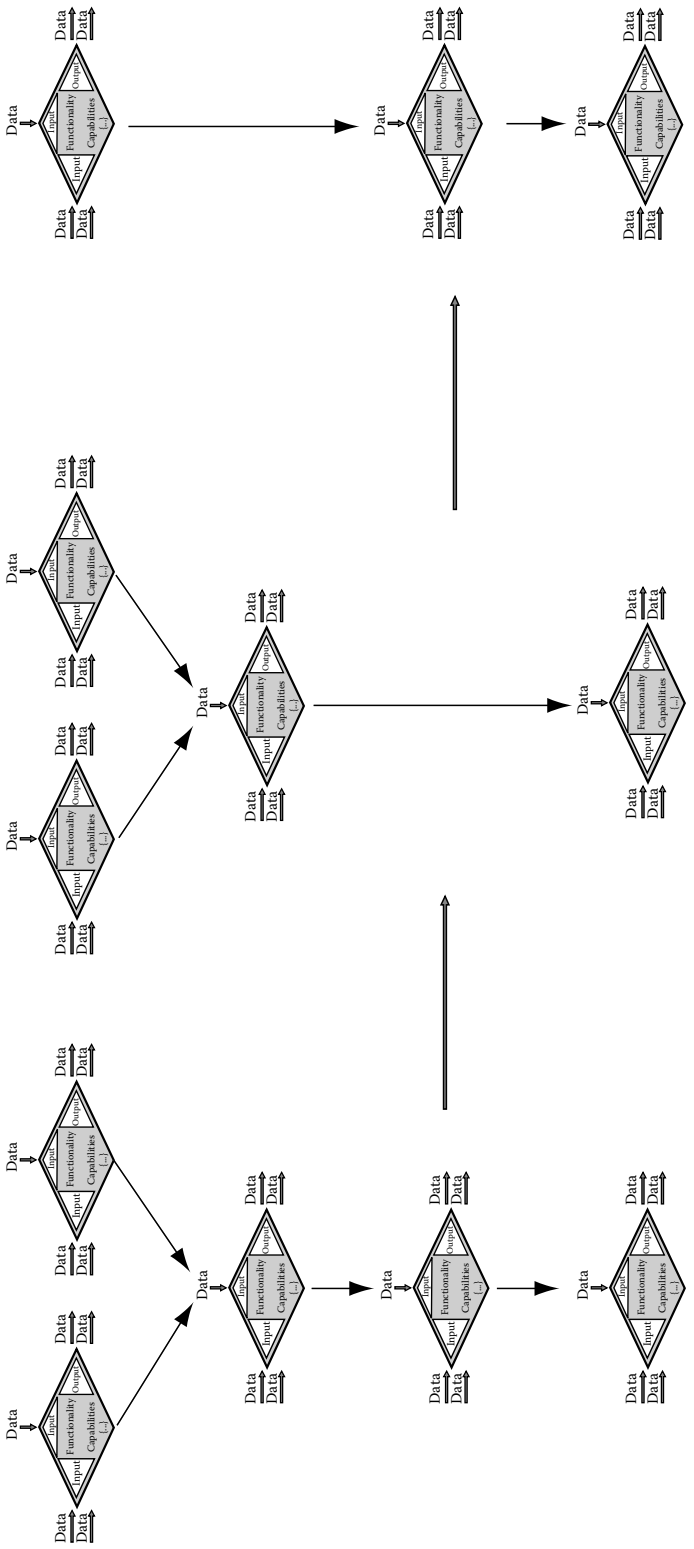


FIGURE 4.7 Architecture transitions.

shared resources in the system) and the name of the data they produce. The scheduler needs all this information to take the decisions.

From the point of view of the operating system, a new component that takes care of all the data exchange needs to be implemented. This would in fact be an extended message-passing mechanism, with the added feature of notifying the scheduler when new data types become available. The mapping of this module in the architecture is the constraint imposed to the protocols to send/receive data via, for example, a publish/subscribe mechanism to the central scheduler.

An efficient naming system for the entities and the data is needed. Downloading new entities to a sensor node involves issues similar to services discovery. Several entities with the same functionality but with different requirements and capabilities might coexist. The data-centric scheduler has to make the decision which one is the best.

The architecture presented earlier might be extended to groups of sensor nodes. Several data-centric schedulers together with a small, fixed number of protocols can communicate with each other and form a virtual backbone of the network [14].

Entities running inside sensor nodes can be activated using data types that become available at other sensor nodes (for example, imagine one node using his neighbor routing entity because it needs the memory to process some other data).

Of course, this approach raises new challenges. A naming system for the functionality and data types, reliability issues of the system (for factors such as mobility, communication failures, node failures, and security attacks), etc. are just a few examples. Related work on these topics already exist (e.g., [15,16]). In the case of the data-centric architecture we propose, this aspect is a research topic addressed by the Featherlight research project [17].

4.3.4 Generalized Component Interfaces

Sensor networks are a relatively new field of science. A large number of prototypes have been already built and some commercial application already exist. One of the main problems a system integrator faces at this moment is the lack of standards and basically the lack of a standardized interface to the data available via the sensor networks.

The lack of standardized interfaces manifests itself at various levels: the interface between application and the architecture, interface between application and networking protocols, interface between the gateway and an external computer, and the interface between the computer and existing infrastructure. The end result of this lack of standardization reflects itself in the usability of sensor networks and their adoption by the general public. Applications are written now mainly for specific platforms and specific networking protocol stacks and sharing of data between various applications needs to be implemented almost from scratch every time.

In this section we will explore one alternative for such an interface. The motivation for this effort is quite straight forward—we can only bring into discussion the fact that interfacing the various building blocks of a sensor network takes more development time then implementing the specific application. The existence of a generally accepted interface will allow various protocols to be actually stacked on top of each other; it will also allow quantification of the behavior of various algorithms/protocols in a given setup. Other benefits of having such an interface would be reducing the number of interfaces needed to access all the data provided by the sensors in the network, reducing the number of packet formats to be transmitted by the MAC and routing protocols, etc.

The starting point for the generalized component interface is the concept of entity itself. As mentioned previously, by entity we understand a basic software component having a dedicated functionality. The entity provides already a number of parameters as the details needed for the real time scheduler, the details about the data it produces and it consumes, etc. We extend this information with a description on the way the actual entities provide functionality to the exterior.

The generalized component interface consists of the following basic concepts:

- *Identification means*: The entity needs to be uniquely identifiable from the user perspective. A tuple of parameters containing the node identifier, the entity identifier, and the sensor instance identifier is proposed as a solution.
- *Functionality access*: The user will interact with the entity by calling a function (eventually receiving the result returned by the function). The entity will provide two sets of functions: a fixed set (generic functions available in every entity as configuration functions, service description, standard access functions, etc.) and a user-specific set (with no restrictions applied to it).
- *Standardized parameters*: The functions provided by the entity need to accept parameters of various types and sizes. We propose to use one standard data structure containing two fields: the size of the user data and the actual payload of the user data for that function. Because the user calls a specific function, it knows what parameters it needs to send and can transform them in a byte array (a "void *" pointer in the C language) of a known length. The function in the entity will cast this array of bytes to the structure it expects as input as the first operation and then perform as any regular function. The return parameters are passed based on the same philosophy. Because both the user and the function have a clear understanding of the data structure for a given function, no additional description needs to be sent in clear.

When building the generalized interface as an application block we can built it out of three small layers:

- *Physical layer*: Any physical interface existing in the node. Examples include the radio interface, the serial port, the digital I/O lines, etc. No specific software needs to be developed here except the hardware abstraction layer (hardware driver) usually provided by the system software.
- *Data-link layer*: This layer is a small extension of the data-link layer existent already on top of the chosen physical layer. The addition consists in specifying the payload to be sent/received by this layer in the form of two fields: the size of the byte array and the byte array itself. This is a straight forward extension on most data-link layers.
- *Standard interface layer*: This layer needs to be developed once for every entity. As we already mentioned, this layer needs to provide a unique mean of identification, a set of predefined functions, and a set of user defined functions. Additional mechanisms (as reporting an error when an nonexistent function was called or when the data structure sent to a function cannot be cast to the needed data structure) need to be set in place.

The main advantage of this approach is the large flexibility offered. At the physical level, it does not really matter though which interface the data arrived to the entity (via the serial port, via the radio, etc.). The data-link layer stays basically unchanged. The standard interface layer allows users/applications to call any function as long as they supply the correct input parameters.

4.3.5 Concluding Remarks

In this chapter we introduced the concept of a flexible data-centric architecture that gathers its strength from combining the concepts of architecture and operating system in one single notion. Additional strength is gained by offering a unified approach in calling functionality across all the entities on all devices in the network.

These concepts have already been tested in simulators and implemented in practice as part of the AmbientRT, a real-time operating system targeted at sensor networks [18]. The theoretical foundation were explored as part of the EYES [1] and Featherlight [17] projects.

4.4 Distributed Data Extraction Techniques

The goal of a sensor network is to provide the user with easy access to the data being collected by the sensors employed in its nodes. A WSN designed to service injected queries generally needs to carry out three main tasks:

1. Inject the query into the network.
2. Sample the sensor specified in the query.
3. Return the acquired data to the sink node that originally injected the query.

In order to maximize the lifetime of the sensor network, it is important to carry out all three tasks mentioned above in an energy-efficient manner. This section provides an overview of some of the important techniques that can be found in current literature that address the tasks listed above. The mechanism of data dissemination is an important part of the architecture of the sensor networks—it describes in fact the way in which this technology can be used.

We begin by first describing several important attributes that may be specified within a query. We illustrate how the techniques found in the recent literature manage the various attributes. Next we highlight the four essential building blocks which make up a distributed data extraction framework for WSNs. We then describe the state of the art for each category and discuss the salient points of the various data extraction methods. A summary of the discussion in Section 4.4.2 can be found in Table 4.1.

4.4.1 Attributes of Queries

Queries form an essential part of any deployed WSN as they are wholly responsible for extracting all the data that is requested by the end-user. Queries may be actively injected into the network once the network has been deployed or they may be predefined within every node prior to deployment. There are a number of attributes that are usually specified in a query and these attributes help decide how and which data needs to be collected. The manner in which the specified query attributes are processed can have a significant impact on the overall network lifetime.

Any query must have three user-defined attributes specified within it: (i) duration, (ii) scope, and (iii) result. Figure 4.8 gives an overview of the various attributes. We now describe every attribute in greater detail.

1. *Duration*: This attribute indicates for how long a particular query needs to return results to the user. Typically, there are two possible options. Either the user requires a snap-shot of the sensor readings in the network at a particular specified time instance or the user needs data to stream in from the sensors periodically for an extended duration. Queries running for an extended duration are referred to as long-running queries. Note that the duration refers to the time period after data collection begins. Thus for event-based queries which request for data only when a particular event occurs, the "duration" attribute refers to the time after the event has occurred.
2. *Scope*: A user may not always require data to be collected from all the sensors in the network at all times. Instead, one may limit the scope of the query by specifying certain spatial, temporal, or sensor-based constraints. Queries with such constraints are known as range queries. For example, readings may be requested only if the sensor readings go

TABLE 4.1 A Summary of Distributed Data Extraction Techniques for WSNs

	Characteristics							
	Essential Conceptual Features				Platform			
Name of the Project/ Mechanism Author	In-Network Processing	Acquisitional Query Processing	Cross-Layer Optimization	Data-Centric Query/ Data Dissemination	Simulation	Mote-Class	PDA-Class	PC-Class
Directed diffusion [9]	Uses filters Suppression of duplicate messages	×	×	Publish/subscribe application programming interface (API) based on named data Gradients used to route data from source to sink		✓	✓	
Cougar [19]	Aggregation at cluster leaders Packet merging	×	×	×			✓	
TinyDB [20]	Evaluation of operators such as MIN, MAX, COUNT, AVERAGE, etc.	Centralized query optimization based on collected metadata Priority given to cheapest sensor when evaluating multiple predicates Multi-query optimization: Grouping of multiple identical events	Communication scheduling primarily for upstream data flow Nodes able to decide on schedule themselves using hop count Minimizes burden on MAC Scheme unusable for multiple concurrent queries with different epoch requirements	Semantic routing Designed for range queries Maintenance algorithm may be too costly if measured attribute changes too rapidly	✓	✓		
Bonfils et al. [23]	Decentralized query operator placement	×	×	×	✓			

(*continued*)

TABLE 4.1 (continued)

Name of the Project/ Mechanism Author	Characteristics							
	Essential Conceptual Features				Platform			
	In-Network Processing	Acquisitional Query Processing	Cross-Layer Optimization	Data-Centric Query/ Data Dissemination	Simulation	Mote-Class	PDA-Class	PC-Class
TiNA [21]	Exploits temporal correlations Readings transmitted only when user-specified threshold is exceeded	×	×	×	✓			
Deligiannakis et al. [22]	Transmits approximated readings generated using base signals	×	×	×	✓			✓
WaveScheduling	×	×	Nodes scheduled to minimize MAC collisions Saves energy, but increases latency	×	✓			
Coman et al. [24]	×	×	×	Queries forwarded based on spatial and temporal ranges	✓			
REED [25]	Joins between static, external tables, and sensor nodes	×	×	×	✓	✓		
BBQ [26]	×	Minimize sampling by predicting sensor readings using centralized statistical models Identify temporal and spatial correlations	×	×	✓			
Ken [27]	Nodes keep local model synchronized with central model Able to detect outliers	×	×	×	✓			

TD-DES [28]	×	×	Event scheduler dynamically allocates and multiplexes upstream and downstream time slots for each event type Root works out schedule	×	✓	
DTA [29]	×	×	DTA defines rules controlling order of data transmission Root works out schedule	×	✓	✓
AI-LMAC [30]	×	×	Adapts operation of MAC based on requirements of application Distributed bandwidth allocation based on expected traffic for a particular query MAC operates with low duty cycle in inactive areas of network	Directs queries only to relevant parts of the network Queries directed based on attribute name and value	✓	
GHT [31]	×	×	×	Hashes attribute name into geographic coordinates Nodes must be location aware Uses perimeter refresh protocol to improve robustness Uses structured replication to eliminate bottlenecks during occurrence of multiple identical events	✓	✓

(*continued*)

TABLE 4.1 (continued)

	Characteristics							
	Essential Conceptual Features				Platform			
Name of the Project/ Mechanism Author	In-Network Processing	Acquisitional Query Processing	Cross-Layer Optimization	Data-Centric Query/ Data Dissemination	Simulation	Mote-Class	PDA-Class	PC-Class
DIFS [32]	×	×	×	Supports range queries Hashes attribute name and value into geographic coordinates Nodes must be location aware Does not deal adequately with node failures and packet losses	✓			
DIM [33]	×	×	×	Supports range queries Hashes event to a zone Nodes must be location aware Reassigns zones when nodes enter or leave ACK scheme improves robustness			✓	✓
DOSA [34]	Aggregates data by taking advantage of correlations of sensor readings of adjacent sensor nodes	×	×	Publish/subscribe API based on named data Gradients used to route data from source to sink		✓	✓	

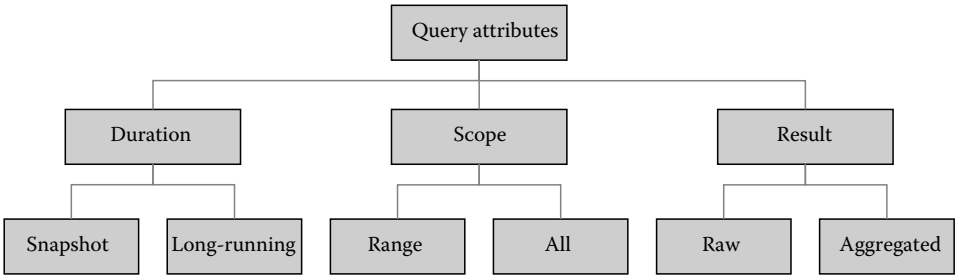


FIGURE 4.8 Attributes specified within a query.

above a certain threshold, or if a sensor is located in a specific area or at a particular time of the day. These constraints may also be specified in various combinations. Note that "scope" may not necessarily be a static parameter such as location or sensor type. It could be a dynamic parameter as well such as temperature. For example, if we consider a static network in an outdoor environment, a query that requires temperature readings from a particular region will always result in readings coming from the same part of the network regardless of the time of the day. However, if a user requires humidity readings only if the temperature goes above a particular threshold, the query might result in different sets of sensors returning readings depending on the time of the day. Event-based queries make use of the "scope" parameter as they make use of the temporal constraint, i.e., instead of returning data all the time, data is sent to the user only when a particular event occurs. Conversely, a user may require all the readings from the network.

3. *Result*: The "result" attribute refers to the format of the data that is requested by the user. The data can either be in "raw" format—in which case the user requires every reading sampled by the sensors chosen by the "scope" attribute, or the user may require aggregated data, e.g., the average of all the chosen sensors.

Table 4.2 gives an overview of how the some of the significant projects/papers in the field of distributed data extraction address the three attributes described above. Note that the table illustrates which attributes the particular method is specifically designed for. For example, we have stated in the table that TinyDB supports snap-shot queries. However, in reality, TinyDB can also accept long-running queries. But the optimizations built into TinyDB are specifically geared toward snap-shot

TABLE 4.2 A Summary of How Query Attributes Are Managed by Different Projects

	Duration		Scope		Result	
Name of the Project/Mechanism/Author	Snapshot	Long-Running	Range	All	Raw	Aggregated
Directed diffusion [9]		✓	✓			✓
Cougar [19]	✓		✓			✓
TinyDB [20]	✓		✓			✓
TiNA [21]	✓		✓			✓
Deligiannakis et al. [22]		✓		✓	✓	
Bonfils et al. [23]	✓		✓			✓
Coman et al. [24]		✓	✓			
REED [25]	✓		✓		✓	
BBQ [26]		✓		✓	✓	
Ken [27]		✓		✓	✓	
TD-DES [28]		✓	✓		✓	
DTA [29]		✓	✓		✓	
AI-LMAC [30]		✓	✓		✓	
GHT [31]	✓		✓			
DIFS [32]	✓		✓			
DIM [33]	✓		✓			
DOSA [34]		✓		✓	✓	
PAQ [35]		✓		✓	✓	
Chatterjea et al. [36]		✓		✓	✓	

queries. We advise the reader to refer to this table while reading Section 4.4.2 which provides details of the various projects/papers.

4.4.2 Essential Conceptual Building Blocks

While there are a host of features that are necessary to form a full-fledged distributed data extraction system for WSNs, we highlight four essential conceptual building blocks in Figure 4.9 that have been widely mentioned in the existing literature. Below, we provide a very brief overview of each building block and describe their specific roles. This is followed by a discussion detailing how the various components are related to each other thus enabling the reader to visualize how the various components fit into the "bigger picture."

- *In-network processing*: This involves moving various types of computation that are traditionally done on the server-side to within the sensor network itself. It is generally used for filtering and processing of messages flowing within the network thus preventing the transmission of unnecessary information.
- *Acquisitional query processing*: Energy consumption in sensor nodes depends on two main factors: Operation of the transceiver and operation of the sensors. Acquisitional query processing helps minimize energy consumption by targeting the sensors, i.e., sampling of the various attached sensors is carried out in an energy-efficient manner. For example, a user may be presented with sensor readings generated using certain statistical methods rather than actually sampling sensors.
- *Cross-layer optimization*: Unlike conventional computer networks which can generally be used to perform a wide variety of tasks, WSNs are usually designed for a particular application. This makes it possible to design the various components of the WSN architecture, e.g., the routing and MAC protocols specifically for the application in mind. This could mean that the MAC and routing protocols may be able to adapt to the changing requirements of the application. This is fundamentally different from the conventional OSI model used for typical networks where the lower-layer protocols operate completely independently from the higher-layer protocols.
- *Data-centric data/query dissemination*: Unlike conventional routing protocols which do not actually bother about the content of the data message being transmitted, the path taken by a message being routed by a data-centric data/query dissemination protocol for a WSN is completely dependent on the contents of the message. This allows messages to be routed more efficiently.

While Figure 4.9 illustrates the pertinent features of every building block, it does not illustrate the relationship between them. We present this relationship in Figure 4.10. Probably the most noticeable feature is that we have placed acquisitional query processing, cross-layer optimization, and data-centric data/query dissemination all within the class of in-network processing. The reason for this can be traced back to the way data is usually collected in conventional databases using the warehousing approach. Under this model, data is initially collected from various sources (e.g., sensors with wireless transmitters) and stored at a central location. The data is then processed centrally to extract the required information. This model is highly unsuitable for WSNs as it involves excessive transmission overhead and also prevents users from accessing real-time, streaming data. The only viable alternative is to migrate from off-line processing to processing the data within the network as close to the data source as possible. The practice of processing data at the sensor nodes themselves is known as in-network processing and can result in a significant reduction in the number of messages transmitted. The ability to perform in-network processing is a cornerstone of WSNs.

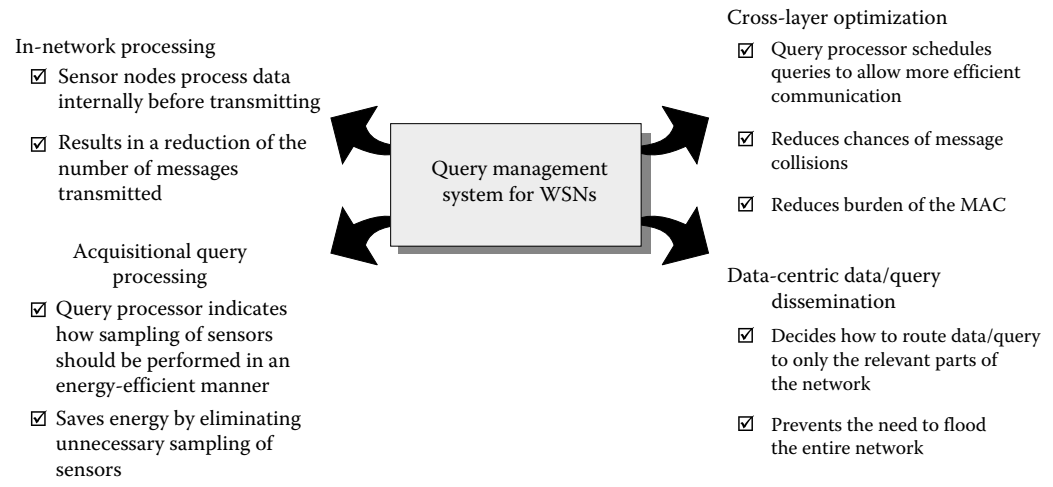

In-network processing
☑ Sensor nodes process data internally before transmitting
☑ Results in a reduction of the number of messages transmitted
Acquisitional query processing
☑ Query processor indicates how sampling of sensors should be performed in an energy-efficient manner
☑ Saves energy by eliminating unnecessary sampling of sensors
Query management system for WSNs
Cross-layer optimization
☑ Query processor schedules queries to allow more efficient communication
☑ Reduces chances of message collisions
☑ Reduces burden of the MAC
Data-centric data/query dissemination
☑ Decides how to route data/query to only the relevant parts of the network
☑ Prevents the need to flood the entire network

FIGURE 4.9 Essential building blocks of a distributed query extraction system for WSNs.

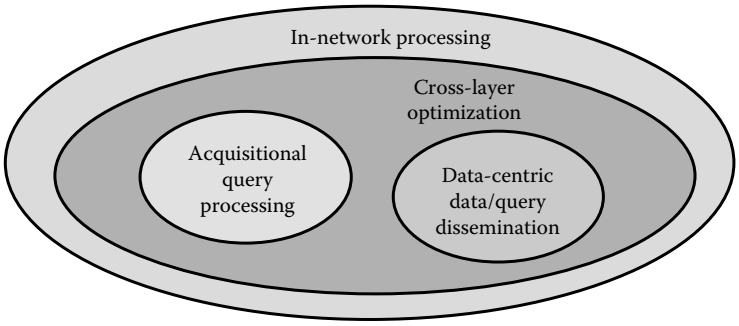


FIGURE 4.10 Relationship between the building blocks.

In order to distinguish between sensors attached with wireless transmitters and wireless sensor nodes, which are actually capable of performing simple computations within them, and also because in-network processing is a very generic term (i.e., it could essentially refer to any sort of processing that takes place within a node) we have placed acquisitional query processing, cross-layer optimization, and data-centric data/query dissemination all within the class of in-network processing. All these three subclasses require a sensor node to analyze the data it has at hand and make certain decisions based on the outcome of the analysis instead of transmitting all the data to a central server for off-line processing or centralized decision-making. Such processing is not possible in normal sensors that only have attached wireless transmitters. The kind of processing that can take place within a wireless sensor node varies greatly, which is why we have these three other subcategories.

Cross-layer optimization generally refers to schemes where the application influences the operation of the routing or MAC layers. In the cross-layer optimization category, we cover literature which studies the relationship between the MAC and the application layers. Data-centric data/query dissemination is classified as a subclass of cross-layer optimization as it involves creating links between the application and routing layers. Acquisitional query processing falls under cross-layer optimization as it deals with schemes where sensor sampling can be dependent on information gathered from the injected query (i.e., application layer) or even the underlying MAC.

In order to simplify matters and also because the features present in a particular class are not present in its super class, we analyze each building block separately in the following subsections. However, the reader should keep in mind the existing relationships indicated in Figure 4.10. Sections 4.4.3 through 4.4.7 describe each category in greater detail and illustrate the different methods that have been employed to incorporate the various features.

4.4.3 In-Network Processing

While the precise manner in which in-network processing is carried out on sensor nodes may differ between various publications, the fundamental objective is still the same—to save energy by reducing message transmissions. Directed diffusion [9] performs in-network processing using filters. When a node receives a message that matches its filter, the message is first handed over to the application module within the node for processing instead of forwarding it to the next node. For example, the application might carry out a suppression of duplicate messages indicating the detection of the occurrence of an event so as to prevent a sudden burst of identical messages when a bunch of nodes detect the same event. The main drawback of directed diffusion however, is that it has a nondeclarative interface and thus does not rank very highly in terms of usability. As shown in Table 4.2, directed diffusion is designed primarily to extract aggregated data from a certain part of the network over an extended period of time.

Cougar [19], which was one of the first projects to view a sensor network as a distributed database with a declarative interface, uses a clustered approach to in-network processing. A network may consist of several clusters each of which is made up of a single leader node and a group of child sensor nodes belonging to the leader. Child nodes periodically send their readings to the leader node which then aggregates the received readings and only continues to forward the computed result toward the root of the network. Computation at the leader only takes place once all the child nodes have responded. Additionally, since sending multiple small packets is more expensive than sending one larger packet (due to the packet header payload and the cost of reserving the medium) Cougar performs packet merging by combining several packets into one. This method is particularly beneficial when servicing queries that generate holistic aggregates where intermediate nodes cannot perform any partial aggregation and all data must be brought together to be aggregated by the node evaluating the query, e.g., the Median operator or even the collection of raw sensor readings. It is important to note, however, that while Cougar claims to be designed for WSNs, it was deployed on PDA-class devices that had significantly larger processing power and could even run Windows CE and Linux [20]. Their design does not consider the impoverished power and computational constraints of conventional sensor nodes, e.g., XML, which is known for its verbosity, is used to encode messages. Apart from packet merging, Cougar lacks any other features that make it energy-efficient. It also fails to address the issues of reliability and self-organization and relies instead on the underlying 802.11 MAC protocol.

TinyDB [20] supports a number of aggregation operations (e.g., MIN, MAX, SUM, COUNT, AVERAGE, etc.) over certain user-specified sample intervals. As sensor readings flow up the communication tree, they are aggregated by intermediate nodes that are able to meet the requirements of the query (Figure 4.11). Without aggregation, every node in the network needs to transmit not only its own reading but also those of all its children. This causes a bottleneck close to the root node and also results in unequal energy consumption, i.e., the closer a node is to the root node, the larger the number of messages it needs to transmit which naturally results in higher energy consumption. Thus nodes closer to the root node die earlier. Losing nodes closer to the root node can have disastrous consequences on the network due to network partitioning. Using in-network aggregation however, every intermediate node aggregates its own reading with that of its children and eventually transmits only one combined result. This also naturally implies that the size of the message remains constant as it traverses from the source nodes to the root. TinyDB also illustrates how aggregation can be

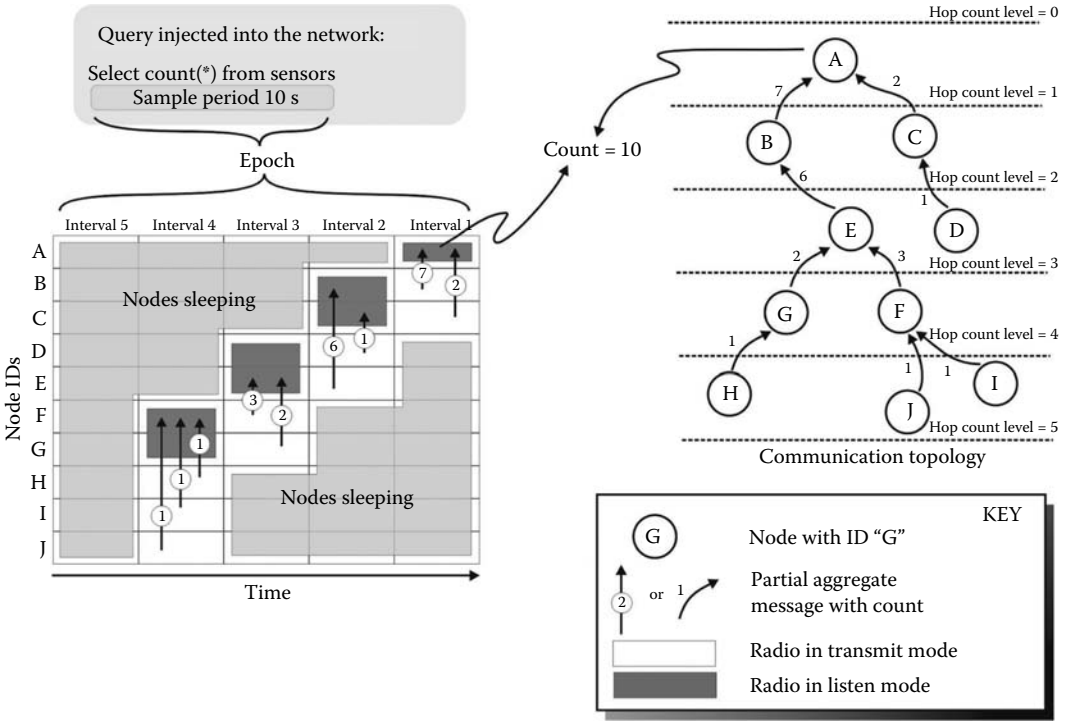


FIGURE 4.11 Aggregation operation using interval-based communication scheduling.

extended to perform more complex tasks such as vehicle tracking and isobar mapping [37]. TinyDB depends on MAC-level acknowledgments and runs a tree-maintenance algorithm continuously in order to ensure that communication between nodes can continue seamlessly at all times.

The optimizations found in Cougar, TinyDB, and TiNA are suitable for mainly snapshot, range queries where the user is interested in aggregated results, e.g., MIN, MAX, SUM, AVG, etc. On the other hand, it is possible for these mechanisms to acquire raw data over extended periods of time, e.g., in all these approaches, the user could simply inject a `SELECT *` SQL-like query into the network. However, none of the mechanisms perform any optimizations to efficiently gather raw data over extended time periods.

The distributed and self-organizing scheduling algorithm (DOSA) for energy-efficient data aggregation [34] scheme helps reduce transmissions by taking advantage of the spatial correlations that might exist between neighboring sensor nodes. The authors present a self-stabilizing scheduling algorithm that decides when certain nodes in the network should act as correlating nodes. Correlating nodes are in charge of identifying any correlations that may exist between their own readings and readings of nodes in their one-hop neighborhood. If correlations exist, a correlating node transmits this correlation information to the sink and also its own sensor readings. The sink node then uses the correlating node's own reading and combines the received correlation information to compute the readings of the correlating node's one-hop neighbors. The authors not only present some theoretical bounds on the energy consumption but also implement the algorithm on actual sensor nodes to demonstrate that how it functions in real-life. However, due to its cross-layered design, the a node's scheduling scheme may be adversely affected if has a poor link quality with its one-hop neighbors.

While the majority of the literature in this category focuses on utilizing different strategies to ensure energy-efficient operation, none of the authors investigate the performance of unreliable message

transmissions. As current sensor nodes are known to be unreliable and the performance of the various in-network processing schemes can change dramatically with varying degrees of reliability, this remains as a topic which is open to further research.

4.4.4 Acquisitional Query Processing

In conventional database systems, when posing a query to the system, a user need not bother about how the data should be extracted in the most efficient way. The same concept holds true in distributed database systems where the requested data might be stored in small fragments spanning the entire network. In this case, the user does not require knowledge of where the data resides. In other words, the techniques used to locate data or the methods followed to extract data in an efficient manner, are processes that are completely transparent to the user. The user simply injects the query into the network and waits for the result to appear. It is the responsibility of the query processing system to handle the above-mentioned tasks and act as an interface between the user and the data sources.

One of the tasks of a query processor for WSNs is to generate an optimized query execution plan that outlines an energy-efficient strategy to execute a query. While conventional database query optimization techniques calculate the cost of executing a query based on a number of parameters such as CPU instructions, I/O operations, messages transmitted, etc. the model necessary for WSNs is slightly different due to its unique characteristics described earlier. The tight power constraints of WSNs have driven researchers to find novel ways to minimize the two main sources of power consumption on a sensor node—the operation of the radio transceiver and the sampling of the sensors to obtain readings. The idea of managing sensor sampling as one of the tasks of the query processor was first introduced in TinyDB [20] and was termed as acquisitional query processing.

TinyDB carries out query optimization at the root node. Metadata such as the energy and time required to sample a particular sensor, information about the costs of processing and delivering data, etc. are periodically copied from the nodes to the root for use by the query optimizer. The optimizer also has the task of initializing, merging, and updating the final value of partial aggregate records as they propagate through the network toward the root.

Before a query is injected into the network through the root node, the query optimizer can optimize the query in various ways using the collected metadata. For example, since the difference in power consumption of sampling various sensors on a single node can differ by several orders of magnitude, the order in which sensors are sampled when evaluating a query can have a substantial impact on network lifetime.

Consider the scenario where a user requires readings from a light sensor when the temperature and humidity sensors reach thresholds t and h, respectively. If the node uses a humidity sensor that requires 100 times more energy to sample than the temperature sensor, in most cases, it would be a lot more energy-efficient to sample the temperature sensor first to check if the threshold has been met and proceed with the sampling of the humidity sensor only if the result of the temperature threshold check is found to be true. In fact, in such a scenario, sampling the humidity sensor first could be up to an order of magnitude more expensive.

TinyDB also performs multi-query optimization on event-based queries in order to reduce costs due to sensor sampling and transmission. Consider a query Q that requests temperature readings only when a certain event E occurs. The occurrence of a string of event Es within a short time interval would trigger multiple instances of the query to run simultaneously. This results in high energy consumption as every instance of a query performs its own sensor sampling and transmission of results. To alleviate this problem TinyDB optimize's such queries by rewriting them so that all occurrences of event E of the last k seconds are buffered. When a sensor reading is obtained, it is compared against the buffered events and the temperature readings are returned. Thus no matter how frequently events of type E are triggered, only one query is required to run.

4.4.5 Cross-Layer Optimization

Traditional system architectures are commonly known to adhere to the layered protocol stack design where every layer operates in a completely independent manner. The Internet browser on a PC, for example, does not require any knowledge about the kind of network connectivity available. It works regardless of whether the user is connected via Ethernet or 802.11b. Such a layered approach however, is not optimal from the energy efficiency point of view especially when considering WSNs. This is because unlike the network being used in an office LAN, WSNs are typically application-specific networks. Thus it only makes sense to try and make the various components of the WSN architecture more application-aware by performing certain cross-layer optimizations thereby helping to improve network lifetime. Note however, that while the term "cross-layer optimization" in the field of sensor networks might refer to the optimization between the application and routing layers for instance, in this case we refer to the more radical approach where optimization is performed between the application and MAC layer.

As usage of the transceiver is an energy-consuming task, it is imperative that maximum benefit is derived during the time it is operational. Thus rather than encountering energy-wasting collisions during data transmission or actively waiting for messages that do not arrive, current cross-layer optimization techniques use a variety of methods to try and schedule tasks in an energy-efficient manner. We now review a number of these techniques.

TinyDB [20] uses an interval-based communication scheduling protocol to collect data where parent and child nodes receive and send data (respectively) in the tree-based communication protocol. The cross-layer optimization in TinyDB involves (i) reducing the burden on the MAC using specifications from the injected query and (ii) routing data from the source nodes to the root. Each node is assumed to produce exactly one result per epoch (time between consecutive samples), which must be forwarded all the way to the base station. As shown in Figure 4.11, every epoch is divided into a number of fixed-length intervals which is dependent on the depth of the tree. The intervals are numbered in reverse order such that interval 1 is the last interval in the epoch. Every node in the network is assigned to a specific interval which correlates to its depth in the routing tree. Thus for instance if a particular node is two hops away from the root node, it is assigned the second interval. During its own interval, a node performs the necessary computation, transmits its result and goes back to sleep. In the interval preceding its own, a node sets its radio to "listen" mode collecting results from its child nodes. Thus data flows up the tree in a staggered manner eventually reaching the root node during interval 1.

While TinyDB's slotted scheduling protocol does help conserve energy by keeping nodes asleep a significant proportion of time, it is primarily designed for servicing a single query posed to the entire network. The scheme is unusable if there are multiple concurrent queries with different epoch requirements.

Chatterjea et al. [30] describe an Adaptive, Information-centric, and Lightweight MAC (AI-LMAC) protocol for WSNs that adapts its operation based on the requirements of the application. The amount of bandwidth that is allocated to a node is proportional to the amount of data that is expected to flow through it in response to the query it is servicing. Bandwidth allocation is done in a distributed manner and is not static but changes depending on the injected query. Information about the expected data traffic through a node is obtained using a completely localized data management framework that helps capture information about traffic patterns in the network. A major advantage of this approach is that the MAC protocol reduces its duty cycle on nodes that are not taking part in servicing the query, thus improving energy efficiency and limiting communication activity only to areas of the network where it is actually required. The data management framework is also used for efficient query dissemination (i.e., directing queries to only the relevant parts of the network) and query optimization. Thus cross-layer optimization in AI-LMAC addresses the entire spectrum of data management issues, i.e., operation of the MAC, routing, and query optimization.

4.4.6 Data-Centric Data/Query Dissemination

Deciding how to route or disseminate data within a communication network is typically performed using IP-style communication where nodes are identified by their endpoints which can be directly addressed using an address that is unique to the entire network. Such addressing schemes are used in the Internet or office LAN. Due to their application-specific nature however, WSNs tend to use a data-centric addressing scheme where "names" are used to create an abstraction of node network addresses. For example, instead of requesting the reading of a node with a particular ID, a typical query usually requests for an attribute of a phenomenon instead. Data dissemination schemes generally address three main issues:

- How queries are routed only to the relevant nodes from the node injecting the query into the network (Flooding a query to all nodes is not energy-efficient)
- How results are routed back to the querying node
- Robustness: how the scheme copes with dynamic network topologies, i.e., node failures and node mobility

We review a number of data dissemination schemes which use distributed techniques to ensure that queries only propagate towards sensors capable of serving the injected queries.

Directed diffusion [9] is one of the pioneering data-centric data dissemination paradigms developed specifically for WSNs. It is based on a publish/subscribe API where the sink injects an interest (e.g., interest for a particular type of named data, such as the "Detection of a bird") into the network. Every receiving node keeps a copy of the interest in its cache. The entry in the cache also stores a gradient that indicates the identity of the neighboring node that originally sent the interest. Gradients are formed gradually at every node as the interest propagates from one node to another eventually flooding the entire network (Figure 4.12). When a source node is able to service the interest, it sends data back to the sink along the path of the gradients set up initially by the interest. In the event data starts flowing toward the sink along multiple gradient paths, the source node reinforces one or

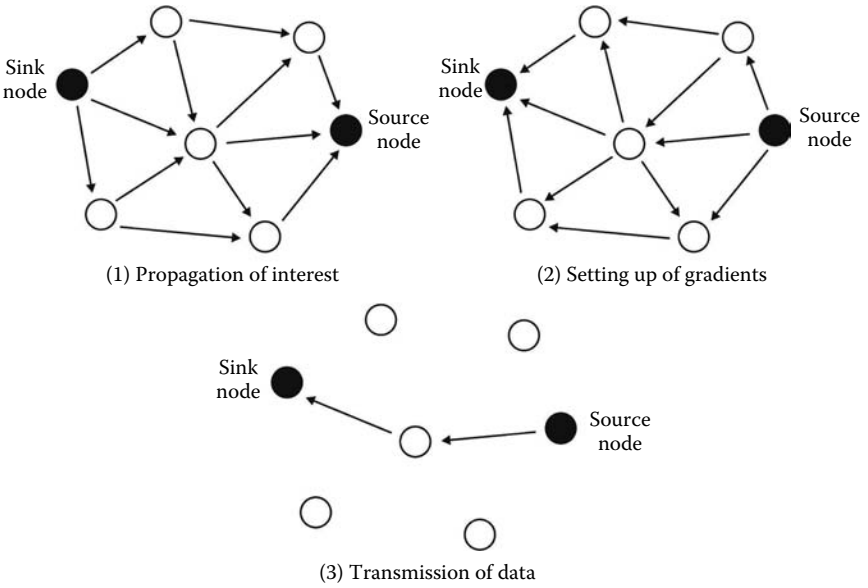


FIGURE 4.12 Setting up of gradients in directed diffusion.

a subset of these paths. Reliable paths are usually reinforced while unreliable ones are removed by expiration due to lack of reinforcements or explicit negative reinforcements. Such gradients allow the local repair of failed or degraded paths and do not require the reflooding of the interest. However, it is necessary to perform flooding when a new interest is injected into the network.

While directed diffusion performs routing based on named data, TinyDB performs routing using a semantic routing tree (SRT) which is based on the actual values of sensor readings. It is useful for servicing range queries. An SRT is an index built over some constant attribute *A* and is stored locally at every node in the network. The index at a node consists of a one-dimensional interval representing the range of *A* values being generated not just by the node itself but also by all its descendants. When a node encounters a query, it only forwards it to its immediate children which are reported to be transmitting values matching the required range specified in the query. The readings may have been generated either by any of the immediate children or by any of the nodes within the subtrees rooted at the immediate children. Additionally, a node executes a query by itself if it can be serviced locally and subsequently transmits the result to its parent. The result eventually propagates up the tree toward the root. If the query cannot be serviced by the node or any of its children, it is dropped. The entry of a child node expires from the SRT of a parent node if the parent node does not receive any updates from the child within a predefined timeout period. The parent then updates its interval information by querying its children and also informs nodes higher up the hierarchy if any changes are detected. While the SRT maintenance algorithm is capable of reflecting changes in the network dynamics (e.g., death of a node), the cost of updating ranges could be prohibitive if the value of the measured attribute changes too rapidly.

Query and data dissemination schemes that prevent the need to flood the entire network have progressed markedly in the recent past. From initially just considering event types or ranges of actual sensor readings, they currently support multiple range queries and use various hashing functions to direct queries to the appropriate sections of the network using the attribute types and values as inputs. Furthermore, they incorporate in-network storage as an integral part of the query dissemination mechanism. However, these newer schemes assume that all nodes are location aware thus increasing the complexity of the system.

4.4.7 Concluding Remarks

As time progresses, WSN deployments are gradually going to grow larger and certain deployments may even be enlarged in stages. This makes it increasingly necessary to improve support for heterogeneous networks, multiple roots and optimization of multiple simultaneous queries that may overlap partially over sensor types, readings, and spatial and temporal parameters. Cross-layer optimizations from the application layer also need to dig in deeper into the network layers and attempt to eventually influence the operation of the MAC. Query optimizations need to be pushed into the network to prevent large amounts of metadata being sent back to the root.

4.5 Conclusion

In this chapter we have outlined the characteristics of WSNs from an architectural point of view. As sensor networks are designed for specific applications, there is no precise architecture to fit them all but rather a common set of characteristics that can be taken as a starting point.

The combination of the data-centric features of sensor networks and the need to have a dynamic reconfigurable structure has led to a new architecture that provides enhanced capabilities than the existing ones. We highlighted the characteristics of the new architecture—for a more detailed description, the user is referred to Refs. [17,38].

The last part of the chapter was dedicated to the mean in which data can be extracted in an efficient way from the sensor network. An interesting observation from the summary presented in Table 4.1 shows that only a handful of the projects have actually been implemented on sensor node platforms. This clearly reflects that while certain projects have begun making inroads into the various essential conceptual features mentioned in this chapter, current distributed data extraction techniques only touch the tip of the iceberg.

References

1. Consortium: Eyes European project. http://eyes.eu.org (2006).
2. Chu, P., Lo, N., Berg, E., and Pister, K.: Optical communication using micro corner cuber reflectors. In: *MEMS97*, Nagoya, Japan (1997), pp. 350–355.
3. Havinga, P.: Eyes deliverable 1.1-system architecture specification (2006).
4. Kahn, J.M., Katz, R.H., and Pister, K.S.J.: Next century challenges: Mobile networking for smart dust. In: *MobiCom '99: Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, New York, ACM (1999), pp. 271–278.
5. Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E.: A survey on sensor networks. *IEEE Communication Magazine* **40** (2002) 102–114.
6. Gislason, D.: *ZigBee Wireless Networking*. CMP Books, Gilroy, CA (2007).
7. TexasInstruments: System-on-chip for 2.4 ghz zigbee/ ieee 802.15.4 with location engine (2007).
8. Incel, O.D., Jansen, P.G., Dulman, S.O., and Mullender, S.J.: Capacity analysis of interfering channels. In: *Proceedings of the 2nd ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks*, Crete, Greece, New York, ACM (2007), pp. 11–18.
9. Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., and Silva, F.: Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking* (TON) **11** (2003) 2–16.
10. Pottie, G.J. and Kaiser, W.J.: Embedding the internet: Wireless integrated network sensors. *Communications of the ACM* **43** (2000) 51–58.
11. VxWorks: Wind river. http://www.windriver.com (2006).
12. Salvo: Pumpkin incorporated. http://www.pumpkininc.com (2006).
13. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D.E., and Pister, K.S.J.: System architecture directions for networked sensors. In: *Architectural Support for Programming Languages and Operating Systems*, New York (2000) 93–104.
14. Gemesi, R., Meratnia, N., and Havinga, P.J.M.: Quality-aware resource management for wireless sensor networks. In Anderson, J.H., Prencipe, G., Wattenhoffer, R., eds.: *Proceedings of the 9th International Conference on Principles of Distributed Systems, OPODIS 2005*, Pisa, Italy. Volume 3974/2006 of *Lecture Notes in Computer Science*, London, Springer Verlag (2005), pp. 412–426.
15. P. Verissimo and Casimiro, A.: Event-driven support of real-time sentient objects. In: *Proceedings of WORDS 2003*, Guadalajara (2003).
16. Cheong, E., Liebman, J., Liu, J., and Zhao, F.: TinyGALS: A programming model for event-driven embedded systems. In: *Proceedings of the 2003 ACM Symposium on Applied Computing*, ACM Press, Melbourne (2003), pp. 698–704.
17. STW-Progress: Feather-light distributed systems project (tes.6388) (2004).
18. Hofmeijer, T.J., Dulman, S.O., Jansen, P.G., and Havinga, P.J.M.: AmbientRT—real time system software support for data centric sensor networks. In: *2nd International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, Melbourne, Australia, Los Alamitos, California, IEEE Computer Society (2004), pp. 61–66.
19. Yao, Y. and Gehrke, J.: The cougar approach to in-network query processing in sensor networks. *SIGMOD Rec.* **31** (2002) 9–18.

20. Madden, S.R., Franklin, M.J., Hellerstein, J.M., and Hong, W.: Tinydb: An acquisitional query processing system for sensor networks. *ACM Transactions Database Systems* **30** (2005) 122–173.
21. Sharaf, A., Beaver, J., Labrinidis, A., and Chrysanthis, K.: Balancing energy efficiency and quality of aggregate data in sensor networks. *The VLDB Journal* **13** (2004) 384–403.
22. Deligiannakis, A., Kotidis, Y., and Roussopoulos, N.: Compressing historical information in sensor networks. In: *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, New York, ACM (2004), pp. 527–538.
23. Bonfils, B.J. and Bonnet, P.: Adaptive and decentralized operator placement for in-network query processing. In: *Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks (IPSN)*. Volume 2634 of Lecture Notes in Computer Science, Springer–Verlag, Berlin Heidelberg (2003), pp. 47–62.
24. Coman, A., Sander, J., and Nascimento, M.A.: An analysis of spatio-temporal query processing in sensor networks. In: *ICDEW '05: Proceedings of the 21st International Conference on Data Engineering Workshops*, Washington, DC, IEEE Computer Society (2005) p. 1190.
25. Abadi, D.J., Madden, S., and Lindner, W.: REED: Robust, efficient filtering and event detection in sensor networks. In: *VLDB '05: Proceedings of the 31st International Conference on Very Large Data Bases*, *VLDB Endowment*, Trondheim (2005), pp. 769–780.
26. Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J.M., and Hong, W.: Model-based approximate querying in sensor networks. *VLDB Journal* **14** (2005) 417–443.
27. Chu, D., Deshpande, A., Hellerstein, J.M., and Hong, W.: Approximate data collection in sensor networks using probabilistic models. In: *ICDE*, Atlanta (2006) p. 48.
28. Cetintemel, U., Flinders, A., and Sun, Y.: Power-efficient data dissemination in wireless sensor networks. In: *MobiDe '03: Proceedings of the 3rd ACM International Workshop on Data Engineering for Wireless and Mobile Access*, New York, ACM (2003), pp. 1–8.
29. Zadorozhny, V.I., Chrysanthis, P.K., and Krishnamurthy, P.: A framework for extending the synergy between mac layer and query optimization in sensor networks. In: *DMSN '04: Proceedings of the 1st International Workshop on Data Management for Sensor Networks*, New York, ACM (2004), pp. 68–77.
30. Chatterjea, S., van Hoesel, L., and Havinga, P.: AI-LMAC: An adaptive, information-centric and lightweight MAC protocol for wireless sensor networks. In: *Proceedings of the 2004 Intelligent Sensors, Sensor Networks and Information Processing Conference* (Dec. 14–17, 2004), pp. 381–388.
31. Ratnasamy, S., Karp, B., Shenker, S., Estrin, D., Govindan, R., Yin, L., and Yu, F.: Data-centric storage in sensornets with GHT, a geographic hash table. *Mobile Networks and Applications* **8** (2003) 427–442.
32. Greenstein, B., Ratnasamy, S., Shenker, S., Govindan, R., and Estrin, D.: DIFS: A distributed index for features in sensor networks. *Ad Hoc Networks* **1** (2003) 333–349.
33. Li, X., Kim, Y.J., Govindan, R., and Hong, W.: Multi-dimensional range queries in sensor networks. In: *SenSys '03: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, New York, ACM (2003), pp. 63–75.
34. Chatterjea, S., Nieberg, T., Meratnia, N., and Havinga, P.: A distributed and self-organizing scheduling algorithm for energy-efficient data aggregation in wireless sensor networks. In: *ACM TOSN*, New York (2008).
35. Tulone, D. and Madden, S.: PAQ: Time series forecasting for approximate query answering in sensor networks. In: *EWSN*, Zurich (2006), pp. 21–37.
36. Chatterjea, S. and Havinga, P.: An adaptive and autonomous sensor sampling frequency control scheme for energy-efficient data acquisition in wireless sensor networks. In: *DCOSS*, Santorini, (2008) pp. 60–78.
37. Hellerstein, J.M., Hong, W., Madden, S., and Stanek, K.: Beyond average: Toward sophisticated sensing with queries. In: *IPSN*, Palo Alto (2003), pp. 63–79.
38. Dulman, S.O.: Data-centric architecture for wireless sensor networks. PhD thesis, Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Enschede, the Netherlands (2005).

5

Overview of Time Synchronization Issues in Sensor Networks

5.1 Introduction 5-1
5.2 Design Challenges 5-2
5.3 Factors Influencing Time Synchronization 5-3
5.4 Basics of Time Synchronization 5-3
5.5 Time Synchronization Protocols for Sensor Networks 5-6
5.6 Conclusions 5-11
References 5-11

Weilian Su
Naval Postgraduate School

5.1 Introduction

In the near future, small intelligent devices will be deployed in homes, plantations, oceans, rivers, streets, and highways to monitor the environment [1]. Events such as target tracking, speed estimating, and ocean current monitoring require the knowledge of time between sensor nodes that detect the events. In addition, sensor nodes may have to time-stamp data packets for security reasons. With a common view of time, voice and video data from different sensor nodes can be fused and displayed in a meaningful way at the sink. Also, medium access scheme such as time division multiple access (TDMA) requires the nodes to be synchronized, so the nodes can be turned off to save energy.

The purpose of any time synchronization technique is to maintain a similar time within a certain tolerance throughout the lifetime of the network or among a specific set of nodes in the network. Combining with the criteria that sensor nodes have to be energy-efficient, low-cost, and small in a multihop environment, this requirement makes a challenging problem to solve. In addition, the sensor nodes may be left unattended for a long period of time, e.g., in deep space or on an ocean floor. When messages are exchanged using short-distance multihop broadcast, the software and medium access time and the variation of the access time may contribute the most in time fluctuations and differences in the path delays. Also, the time difference between sensor nodes may be significant over time due to the drifting effect of the local clocks.

In this chapter, the backgrounds of time synchronization are provided to enable new developments or enhancements of timing techniques for the sensor networks. The design challenges and factors influencing time synchronization are described in Sections 5.2 and 5.3, respectively. In addition, the basics of time synchronization for sensor networks are explained in Section 5.4. Afterwards, different types of timing techniques are discussed in Section 5.5. Lastly, the chapter is concluded in Section 5.6.

5.2 Design Challenges

In the future, many low-end sensor nodes will be deployed to minimize the cost of the sensor networks. These nodes may work collaboratively together to provide time synchronization for the whole sensor network. The precision of the synchronized clocks depends on the needs of the applications. For example, a sensor network requiring TDMA service may require microseconds difference among the neighbor nodes while a data gathering application for sensor networks requires only milliseconds of precision.

As sensor networks are application driven, the design challenges of a time synchronization protocol are also dictated by the application. These challenges are to provide an overall guideline and requirement when considering the features of a time synchronization protocol for sensor networks; they are robust, energy aware, server-less, lightweight, and tunable service.

- *Robust*: Sensor nodes may fail, and the failures should not have significant effect on the time synchronization error. If sensor nodes depend on a specific master to synchronize their clocks, a failure or anomaly of the master's clock may create a cascade effect that nodes in the network may become unsynchronized. So, a time synchronization protocol has to handle the unexpected or periodic failures of the sensor nodes. If failures do occur, the errors caused by these failures should not be propagated throughout the network.
- *Energy aware*: Since each node is battery-limited, the use of resources should be evenly spread and controlled. A time synchronization protocol should use the minimum number of messages to synchronize the nodes in the earliest time. In addition, the load for time synchronization should be shared, so some nodes in the network do not fail earlier than others. If some parts of the network fail earlier than others, the partitioned networks may drift apart from each other and become unsynchronized.
- *Server-less*: A precise time server may not be available. In addition, the time servers may fail when placed in the sensor field. As a result, sensor nodes should be able to synchronize to a common time without the precise time servers. When the precise time servers are available, the quality of the synchronized clocks as well as the time to synchronize the clocks of the network should be much better. This server-less feature also helps to address the robustness challenge as stated earlier.
- *Lightweight*: The complexity of the time synchronization protocol has to be low in order to be programmed into the sensor nodes. Besides being energy-limited, the sensor nodes are memory limited as well. The synchronization protocol may be programmed into a FPGA or designed into an ASIC. By having the time synchronization protocol tightly integrated with the hardware, the delay and variation of the processing may be smaller. With the increase of precision, the cost of a sensor node is higher.
- *Tunable service*: Some services such as medium access may require time synchronization to be always ON while others only need it when there is an event. Since time synchronization can consume a lot of energy, a tunable time synchronization service is applicable for some applications. Nevertheless, there are needs for both type of synchronization protocols.

The above challenges provide a guideline for developing various types of time synchronization protocols that are applicable to the sensor networks. A time synchronization protocol may have a mixture of these design features. In addition, some applications in the sensor networks may not require the time synchronization protocol to meet all these requirements. For example, a data gathering application may require the tunable service and lightweight features more than the server-less capability. The tunable service and lightweight features allow the application to gather precise data when the users require it. In addition, the nodes that are not part of this data gathering process may

not have to be synchronized. Also, the precision of the time does not need to be high, because the users may only need milliseconds precision to satisfy their needs.

As these design challenges are important for guiding the development of a time synchronization protocol, the influencing factors that affect the quality of the synchronized clocks have to be discussed. Although the influencing factors are similar to existing distributed computer system, they are at different extreme levels. These influencing factors are discussed in Section 5.3.

5.3 Factors Influencing Time Synchronization

Regardless of the design challenges that a time synchronization protocol wants to address, the protocol still needs to address the inherent problems of time synchronization. In addition, small and low-end sensor nodes may exhibit device behaviors that may be much worse than large systems such as personal computers (PCs). As a result, time synchronization with these nodes present a different set of problems. Some of the factors influencing time synchronization in large systems also apply to sensor networks [10]; they are temperature, phase noise, frequency noise, asymmetric delays, and clock glitches.

- *Temperature*: Since sensor nodes are deployed in various places, the temperature variation throughout the day may cause the clock to speed up or slow down. For a typical PC, the clock drifts few parts per million during the day [15]. For low-end sensor nodes, the drifting may be even worse.
- *Phase noise*: Some of the causes of phase noise are due to access fluctuation at the hardware interface, response variation of the operating system to interrupts, and jitter in the network delay. The jitter in the network delay may be due to medium access and queueing delays.
- *Frequency noise*: The frequency noise is due to the unstability of the clock crystal [2]. A low-end crystal may experience large frequency fluctuation, because the frequency spectrum of the crystal has large sidebands on adjacent frequencies. The ρ values for quartz oscillators are between 10^{-4} and 10^{-6} [4].
- *Asymmetric delay*: Since sensor nodes communicate with each other through the wireless medium, the delay of the path from one node to another may be different than the return path. As a result, an asymmetric delay may cause an offset to the clock that cannot be detected by a variance type method [10]. If the asymmetric delay is static, the time offset between any two nodes is also static. The asymmetric delay is bounded by one-half the round-trip time between the two nodes [10].
- *Clock glitches*: Clock glitches are sudden jumps in time. This may be caused by hardware or software anomalies such as frequency and time steps.

Since sensor nodes are randomly deployed and their broadcast ranges are small, the influencing factors may shape the design of the time synchronization protocol. In addition, the links between the sensor nodes may not be reliable. As a result, the influencing factors may have to be addressed differently. In the following section, the basics of time synchronization for sensor networks are discussed.

5.4 Basics of Time Synchronization

As the factors described in Section 5.3 influence the error budget of the synchronized clocks, the purpose of a time synchronization protocol is to minimize the effects of these factors. Before developing a solution to address these factors, some basics of time synchronization for sensor networks need

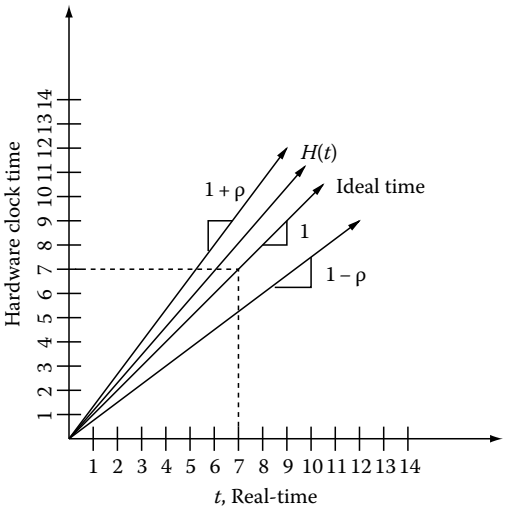


FIGURE 5.1 Drifting of hardware clock time.

to be discussed. These basics are to provide the fundamentals for designing a time synchronization protocol.

If a better clock crystal is used, the drift rate ρ may be much smaller. Usually, the hardware clock time $H(t)$ at real-time t is within an linear envelope of the real-time as illustrated in Figure 5.1. Since the clock drifts away from real-time, the time difference between two events measured with the same hardware clock may have a maximum error of $\rho(b - a)$ [4], where a and b are the time of occurrence of first and second events, respectively. For modern computers, the clock granularity may be negligible, but it may contribute a significant portion to the error budget if the clock of a sensor node is really coarse, running at kHz range instead of MHz. In certain applications, a sensor node may have a volume of cm^3 [18], so a fast oscillator may not be possible or suitable.

Regardless of the clock granularity, the hardware clock time $H(t)$ is usually translated into a virtual clock time by adding an adjustment constant to it. Normally, it is the virtual clock time that we read from a computer. Hence, a time synchronization protocol may adjust the virtual clock time and/or discipline the hardware clock to compensate for the time difference between the clocks of the nodes. Either approach has to deal with the factors influencing time synchronization as described in Section 5.3.

When an application issues a request to obtain the time, the time is returned after a certain delay. This software access delay may fluctuate according to the loading of the system. This type of fluctuation is nondeterministic and may lessen if real-time operation system and hardware architecture are used. For low-end sensor nodes, the software access time may be in the order of few hundred microseconds. For example, a Mica mote is running at 4 MHz [13] having clock granularity of 0.25 μs. If the node is 80% loaded and it takes 100 cycles to obtain the time, the software access time is around 125 μs.

In addition to the software access time, the medium access time also contributes to the nondeterministic delay that a message experiences. If carrier-sense multiple access is used, the back off window size as well as the traffic load affect the medium access time [3,5,21]. Once the sensor node obtains the channel, the transmission and propagation times are pretty deterministic, and they can be estimated by the packet size, transmission rate, and speed of light.

In summary, the delays experienced when sending a message at real-time t_1 and receiving an acknowledgment (ACK) at real-time t_4 are shown in Figure 5.2. The message from node A incurs

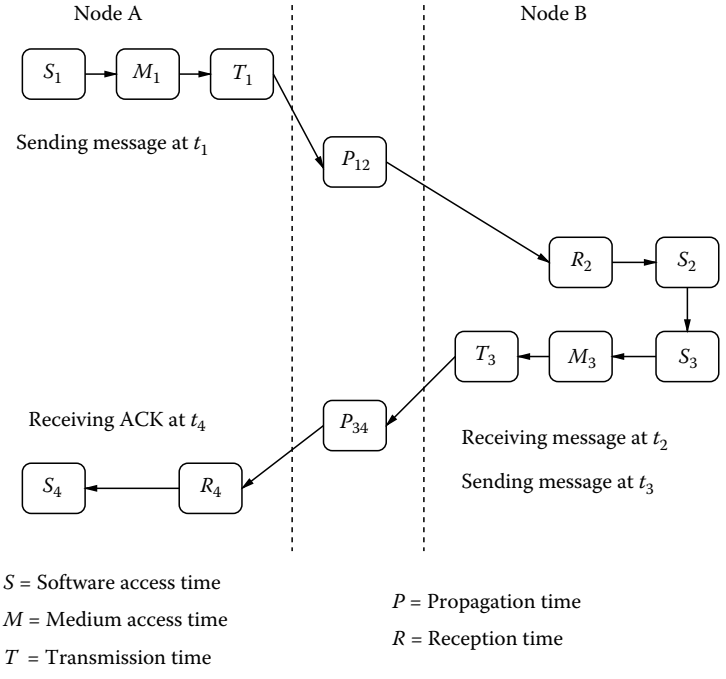


FIGURE 5.2 Round-trip time.

the software access, medium access, transmission, and propagation times. Theses times are represented by S_1, M_1, T_1, and P_{12}. Once the message is received by node B at t_2, it will incur extra delays through receiving and processing. After the message is processed, an ACK is sent to node A at t_3. The total delay at node B is the summation of R_2, S_2, $(1 \pm \rho_B)(t_3 - t_2)$, S_3, M_3, and T_3. After node B sends the ACK, the ACK propagates through the wireless medium and arrives at node A. Afterwards, node A processes the ACK. The path delays for sending and receiving the ACK from node B to A are P_{34}, R_4, and S_4. The round-trip time in real-time t for sending a message and receiving an ACK is calculated by

$$t_4 - t_1 = S_1 + M_1 + T_1 + P_{12} + R_2 + S_2 + (1 \pm \rho_B)(t_3 - t_2) + S_3 + M_3 + T_3 + P_{34} + R_4 + S_4 \quad (5.1)$$

where S, M, T, P, and R are the software access, medium access, transmission, propagation, and reception times, respectively. In addition, ρ_B is the drift rate at node B. The difference $(t_3 - t_2)$ is to account for the waiting time before an ACK is sent to node A by node B.

If the round-trip time is measured using the hardware clock of node A, it has to be adjusted by the drift rate of node A ρ_A. If the granularity of the hardware clock is coarse, the error δ contributed by the granularity should be accounted for. As a result, the round-trip time measured with the hardware clock is bounded by an error associated with the clock drift and granularity as determined by

$$(1 - \rho_A)(t_4 - t_1) \leq H(t_4) - H(t_1) < (1 + \rho_A)(t_4 - t_1) + \delta \quad (5.2)$$

The bound for the round-trip time fluctuates with respect to time since the software and medium access fluctuate according to the load at the node and in the channel. Although the transmission, propagation, and reception times may be deterministic, they may contribute to the asymmetric delay that can cause time offset between nodes A and B.

In the following section, different types of time synchronization protocols are described. Each of them tries to minimize the effect of the nondeterministic and asymmetric delays. For sensor networks, it is best to minimize the propagation delay variation. For example, the delays and jitters

TABLE 5.1 Three Types of Timing Techniques

Type	Description
1. Relies on fixed time servers to synchronize the network	The nodes are synchronized to time servers that are readily available. These time servers are expected to be robust and highly precise.
2. Translates time throughout the network	The time is translated hop-by-hop from the source to the sink. In essence, it is a time translation service.
3. Self-organizes to synchronize the network	The protocol does not depend on specialized time servers. It automatically organizes and determines the master nodes as the temporary time servers.

between two nodes may be different in the forward and return paths. In addition, the jitters may vary significantly due to frequent node failures, since the messages are relayed hop-by-hop between the two nodes. The synchronization protocols in the following section focus on synchronizing nodes hop-by-hop, so the propagation time and variation do not play too much effect on the error of the synchronized clocks. Although the sensor nodes are densely deployed and they can take advantage of the close distance, the medium and software access times may contribute the most in the nondeterministic of the path delay during a one hop synchronization. The way to provide time synchronization for sensor networks may be different for different applications. The current timing techniques that are available for different applications are described in the following section.

5.5 Time Synchronization Protocols for Sensor Networks

There are three types of timing techniques as shown in Table 5.1, and each of these types has to address the design challenges and factors affecting time synchronization as mentioned in Sections 5.2 and 5.3, respectively. In addition, the timing techniques have to address the mapping between the sensor network time and the Internet time, e.g., universal coordinated time. In the following, examples of these types of timing techniques are described, namely the Network Time Protocol (NTP) [14], Timing-sync Protocol for Sensor Networks (TPSN) [8], H-sensor Broadcast Synchronization (HBS) [6], Time Synchronization for High Latency (TSHL) [20], Reference-Broadcast Synchronization (RBS) [7], Adaptive Clock Synchronization [17], Time-Diffusion Synchronization Protocol (TDP) [19], Rate-Based Diffusion Algorithm [11], and Adaptive-Rate Synchronization Protocol (ARSP) [12].

In Internet, the NTP is used to discipline the frequency of each node's oscillator. The accuracy of the NTP synchronization is in the order of milliseconds [9]. It may be useful to use NTP to disciple the oscillators of the sensor nodes, but the connection to the time servers may not be possible because of frequent sensor node failures. In addition, disciplining all the sensor nodes in the sensor field maybe a problem due to interference from the environment and large variation of delay between different parts of the sensor field. The interference can temporarily disjoint the sensor field into multiple smaller fields causing undisciplined clocks among these smaller fields. The NTP protocol may be considered as type (1) of the timing techniques. In addition, it has to be refined in order to address the design challenges presented by the sensor networks.

As of now, the NTP is very computational intensive and requires a precise time server to synchronize the nodes in the network. In addition, it does not take into account of the energy consumption required for time synchronization. As a result, the NTP does not satisfy the energy aware, serverless, and lightweight design challenges of the sensor networks. Although the NTP can be robust, it may suffer large propagation delay when sending timing messages to the time servers. In addition, the nodes are synchronized in a hierarchical manner, and some time servers in the middle of the hierarchy may fail causing unsynchronized nodes in the network. Once these nodes fail, it is hard to reconfigure the network since the hierarchy is manually configured.

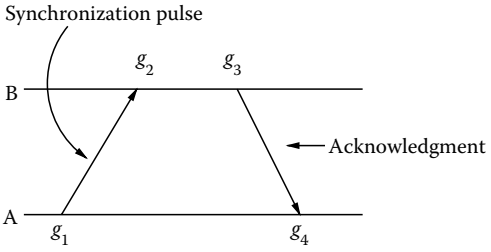


FIGURE 5.3 Two-way message handshake.

Another time synchronization technique that adopts some concepts from NTP is TPSN. The TPSN requires the root node to synchronize all or part of the nodes in the sensor field. The root node synchronizes the nodes in a hierarchical way. Before synchronization, the root node constructs the hierarchy by broadcasting a *level_discovery* packet. The first level of the hierarchy is level 0, which is where the root node resides. The nodes receiving the *level_discovery* packet from the root node are the nodes belonging to level 1. Afterwards, the nodes in level 1 broadcast their *level_discovery* packet, and neighbor nodes receiving the *level_discovery* packet for the first time are the level 2 nodes. This process continues until all the nodes in the sensor field has a level number.

The root node sends a *time_sync* packet to initialize the time synchronization process. Afterwards, the nodes in level 1 synchronize to level 0 by performing the two-way handshake as shown in Figure 5.3. This type of handshake is used by the NTP to synchronize the clocks of distributed computer systems. At the end of the handshake at time g_4, node A obtains the time g_1, g_2, and g_3 from the ACK packet. The time g_2 and g_3 are obtained from the clock of sensor node B while g_1 and g_4 are from the node A. After processing the ACK packet, the node A readjusts its clock by the clock drift value Δ, where $\Delta = (g_2 - g_1) - (g_4 - g_3)/2$. At the same time, the level 2 nodes overhear this message handshake and wait for a random time before synchronizing with level 1 nodes. This synchronization process continues until all the nodes in the network are synchronized. Since TPSN enables time synchronization from one root node, it is type (1) of the timing techniques.

The TPSN is based on a sender–receiver synchronization model, where the receiver synchronizes with the time of the sender according to the two-way message handshake as shown in Figure 5.3. It is trying to provide a lightweight and tunable time synchronization service. On the other hand, it requires a time server and does not address the robust and energy aware design goal. Since the design of TPSN is based on a hierarchical methodology similar to NTP, nodes within the hierarchy may fail and cause nodes to be unsynchronized. In addition, node movements may render the hierarchy useless, because nodes may move out of their levels. Hence, nodes at level i cannot synchronize with nodes at level $i - 1$. Afterwards, synchronization may fail throughout the network.

Currently, there is an interest in improving time synchronization schemes and making them secure from different attacks, such as masquerade, replay, message manipulation, and delay attacks. One such protocol is HBS [6], which is based on TPSN. HBS assumes the cluster heads are high power sensor nodes that can run encryption algorithms and synchronize using GPS. The cluster heads tag each timing message with a sequence number and a message authentication code using the shared key between two nodes. The preloaded public/private keys are protected by tamper-resistant hardware.

To authenticate that a timing message sent by a cluster head is valid, neighboring cluster heads check the message authentication code by using the stored public key. This is assuming that the broadcast by the cluster head is heard by neighboring cluster heads. If the message authentication code is not valid, an alarm message will be broadcasted.

Another variation of the TPSN is TSHL [20], it has two components, clock skew compensation and clock offset correction. For clock skew compensation, the beacon nodes send out beacon messages so nodes can use linear regression to estimate and correct the clock skew. Afterwards, the nodes use

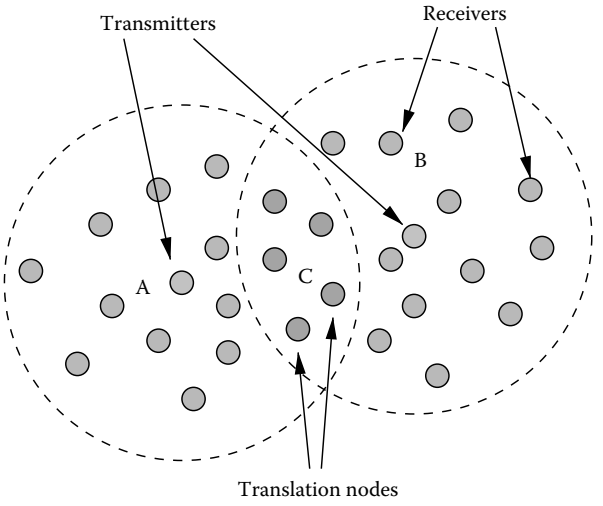


FIGURE 5.4 Illustration of the RBS.

a two-way message handsake as shown in Figure 5.3 to synchronize and correct any clock offset. The objective of TSHL is to address long propagation delay in acoustic sensor networks.

As for type (2) of the timing techniques, the RBS provides an instantaneous time synchronization among a set of receivers that are within the reference broadcast of the transmitter. The transmitter broadcasts m reference packets. Each of the receivers that are within the broadcast range records the time-of-arrival of the reference packets. Afterwards, the receivers communicate with each other to determine the offsets. To provide multihop synchronization, it is proposed to use nodes that are receiving two or more reference broadcasts from different transmitters as translation nodes. These translation nodes are used to translate the time between different broadcast domains.

As shown in Figure 5.4, nodes A, B, and C are the transmitter, receiver, and translation nodes, respectively. The transmitter nodes broadcast their timing messages, and the receiver nodes receive these messages. Afterwards, the receiver nodes synchronize with each other. The sensor nodes that are within the broadcast regions of both transmitter nodes A are the translation nodes. When an event occurs, a message describing the event with a time stamp is translated by the translation nodes when the message is routed back to the sink. Although this time synchronization service is tunable and lightweight, there may not be translation nodes on the route path that the message is relayed. As a result, services may not be available on some routes. In addition, this protocol is not suitable for medium access scheme such as TDMA since the clocks of all the nodes in the network are not adjusted to a common time.

PalChaudhuri et al. [17] extends the work on RBS and provides a probabilistic bound on the accuracy of clock synchronization. The synchronization error is assumed to be Gaussian distributed, and it is possible to calculate the probability of synchronization error given the maximum allowed error. For instance, given the maximum error between two synchronizing nodes is $\epsilon_{\max}$, the probability of synchronization with an error $\epsilon \leq \epsilon_{\max}$ is calculated as follows:

$$P(|\epsilon| \leq \epsilon_{\max}) = 2erf\left(\frac{\sqrt{n}\epsilon_{\max}}{\sigma}\right) \tag{5.3}$$

where

erf is the error function

n is the number of messages needed to achieve the probability P

σ is the standard deviation of the Gaussian distribution

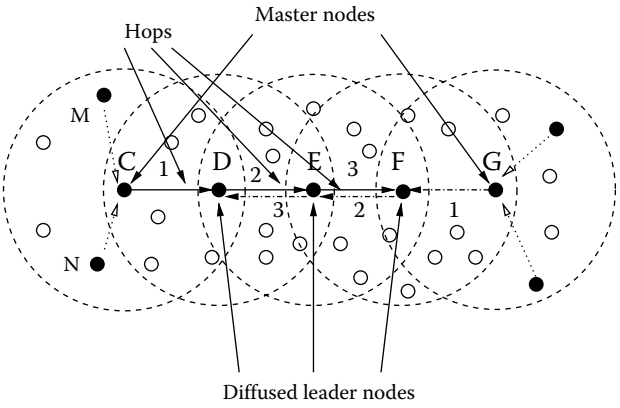


FIGURE 5.5 TDP concept.

Another emerging timing technique is the TDP. The TDP is used to maintain the time throughout the network within a certain tolerance. The tolerance level can be adjusted based on the purpose of the sensor networks. The TDP automatically self-configures by electing master nodes to synchronize the sensor network. In addition, the election process is sensitive to energy requirement as well as the quality of the clocks. The sensor network may be deployed in unattended areas, and the TDP still synchronizes the unattended network to a common time. It is considered as a type (3) of the timing techniques.

The TDP concept is illustrated in Figure 5.5. The elected master nodes are nodes C and G. First, the master nodes send a message to their neighbors to measure the round-trip times. Once the neighbors receive the message, they self-determine if they should become diffuse leader nodes. The ones elected to become diffuse leader nodes reply to the master nodes and start sending a message to measure the round-trip to their neighbors. As shown in Figure 5.5, nodes M, N, and D are the diffused leader nodes of node C. Once the replies are received by the master nodes, the round-trip time and the standard deviation of the round-trip time are calculated. The one-way delay from the master nodes to the neighbor nodes is half of the measured round-trip time. Afterwards, the master nodes send a time-stamped message containing the standard deviation to the neighbor nodes. The time in the time-stamped message is adjusted with the one-way delay. Once the diffuse leader nodes receive the time-stamped message, they broadcast the time-stamped message after adjusting the time, which is in the message, with their measured one-way delay and inserting their standard deviation of the round-trip time. This diffusion process continues for n times, where n is the number of hops from the master nodes. From Figure 5.5, the time is diffused three hops from the master nodes C and G. The nodes D, E, and F are the diffused leader nodes that diffuse the time-stamped messages originated from the master nodes.

For the nodes that have received more than one time-stamped messages originated from different master nodes, they use the standard deviations carried in the time-stamped messages as weighted ratio of their time contribution to the new time. In essence, the nodes weight the times diffused by the master nodes to obtain a new time for them. This process is to provide a smooth time variation between the nodes in the network. The smooth transition is important for some applications such as target tracking and speed estimating.

The master nodes are autonomously elected, so the network is robust to failures. Although some of the nodes may die, there are still other nodes in the network that can self-determine to become master nodes. This feature also enables the network to become server-less if necessary and to reach an equilibrium time. In addition, the master and diffusion leader nodes are self-determined based on their own energy level. Also, the TDP is lightweight, but it may not be as tunable as the RBS.

Li and Rus [11] propose another diffusion technique called rate-based diffusion algorithm. The diffusion algorithm has two types: synchronous and asynchronous. The synchronous diffusion algorithm performs the following steps:

1. Do the following with some given frequency
2. **for** each sensor n_i in the network **do**
3. Exchange clock times with $n_i's$ neighbors
4. **for** each neighbor n_j **do**
5. Let the time difference between n_i and n_j be $t_i - t_j$
6. Change $n_i's$ time to $t_i - r_{ij}(t_i - t_j)$

The synchronous method allows the sensor nodes to exchange their clock values and adjust them by $r_{ij}(t_i - t_j)$, where r_{ij} is the diffusion rate and $(t_i - t_j)$ is the time difference between nodes n_i and n_j. The diffusion rate is greater than zero and can be chosen randomly provided that the $\sum_{j \neq i} r_{ij} \leq 1$. If nodes n_i and n_j are not neighbors, then the diffusion rate is equal to zero.

The synchronous version of the rate-based diffusion algorithm requires all nodes to perform tasks in a set order. To relax such constraint, the asynchronous version of the rate-based diffusion algorithm is executed as follows:

1. **for** each node n_i with uniform probability **do**
2. Ask its neighbors the clock readings (read values from n_i and its neighbors)
3. Average the readings (compute)
4. Send back to the neighbors the new value (write values to n_i and its neighbors)

As with TDP, the Rate-based Diffusion Algorithm is lightweight but cannot provide tunable services.

Macii and Petri [12] propose another adaptive-rate time synchronization technique called ARSP. ARSP is based on the simple response method that adaptively adjust the calibration intervals of measurement instruments based on the last calibration [16]. The objective of ARSP is to assure that the probability of having time offsets larger than $\varepsilon_{\max}$ at the end of kth synchronization is smaller than $1 - P_T$, where P_T is the desired end-of-period synchronization probability.

First, the nodes in the network elects a synchronization master. The synchronization master broadcasts a synchronization packet, and all nodes that receive the packet measure the arrival time, $\widehat{T}_{ik}$. Afterwards, a reference master is elected. The arrival time of the synchronization packet measured by the reference master is $\widehat{T}_{mk}$. The elected reference master sends $\widehat{T}_{mk}$ to all nodes, where they will compare $\widehat{T}_{mk}$ to the arrival time of the synchronization packet. If the difference $\left|\widehat{T}_{ik} - \widehat{T}_{mk}\right|$ is greater than the allowed maximum error $\varepsilon_{\max}$, then the node will set probability p_{ik} to 1; otherwise, p_{ik} is set to 0.

After all the nodes calculate p_{ik}, they send it to the reference master, where it calculates the probability P_k as follows:

$$P_k = \frac{\sum_{i=1, i \neq m}^{M} p_{ik}}{M - 1} \tag{5.4}$$

where

M is the number of sensor nodes that the reference master is connected to

k represents kth synchronization

Afterward, the reference master calculates the next synchronization interval I_{k+1} according to the following:

$$I_{k+1} = \begin{cases} I_k(1 + a) & P_k \geq P_T \\ I_k(1 - b) & P_k < P_T \end{cases} \tag{5.5}$$

where $a > 0$, $0 < b < 1$, and P_T is the desired end-of-period synchronization probability. The ARSP is also lightweight and provides some tunable service such as changing the desired end-of-period synchronization probability and increasing the maximum synchronization error allowed.

In summary, the above mentioned timing techniques may be used for different types of applications; each of them has its own benefits. All of these techniques try to address the factors influencing time synchronization while design according to the challenges as described in Section 5.2. Depending on the types of services required by the applications or the hardware limitation of the sensor nodes, some of these timing techniques may be applied.

5.6 Conclusions

The design challenges and factors influencing time synchronization for sensor networks are described in Sections 5.2 and 5.3, respectively. They are to provide guidelines for developing time synchronization protocols. The requirements of sensor networks are different from traditional distributed computer systems. As a result, new types of timing techniques are required to address the specific needs of the applications. These techniques are described in Section 5.5. Since the range of applications in the sensor networks is wide, new timing techniques are encouraged for different types of applications. This is to provide optimized schemes tailored for unique environments and purposes.

References

1. Akyildiz, I. F. et al., Wireless sensor networks: A survey, *Computer Networks (Elsevier) Journal,* 38, 393–422, March 2002.
2. Allan, D., Time and frequency (time-domain) characterization, estimation, and prediction of precision clocks and oscillators, *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control,* 34(6), 647–654, November 1987.
3. Bianchi, G., Performance analysis of the IEEE 802.11 distributed coordination function, *IEEE Journal on Selected Areas in Communications,* 18(3), 535–547, March 2000.
4. Cristian, F. and Fetzer, C., Probabilistic internal clock synchronization, *Proceedings of the Thirteenth Symposium on Reliable Distributed Systems*, pp. 22–31, Dana Point, CA, October 1994.
5. Crow, B. P. et al., Investigation of the IEEE 802.11 medium access control (MAC) sublayer functions, *IEEE INFOCOM'97,* pp. 126–133, Kobe, Japan, April 1997.
6. Du, X., Guizani, M., Xiao, Y., and Chen, H., *IEEE Global Telecommunications Conference*, pp. 5193–5197, Washington, DC, November 2007.
7. Elson, J., Girod, L., and Estrin, D., Fine-grained network time synchronization using reference broadcasts, *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002),* pp. 147–163, Boston, MA, December 2002.
8. Ganeriwal, S., Kumar, R., and Srivastava, M. B., Timing-sync protocol for sensor networks, *ACM SenSys 2003,* pp. 138–149, Los Angeles, CA, November 2003.
9. IEEE 1588, Standard for a precision clock synchronization protocol for networked measurement and control systems, 2002.
10. Levine, J., Time synchronization over the Internet using an adaptive frequency-locked loop, *IEEE Transaction on Ultrasonics, Ferroelectrics, and Frequency Control,* 46(4), 888–896, July 1999.
11. Li, Q. and Rus, D., Global clock synchronization in sensor networks, *Proceedings of the IEEE Conference on Computer Communications (INFOCOM 2004),* vol. 1, pp. 564–574, Hong Kong, China, March 2004.
12. Macii, D. and Petri, D., An adaptive-rate time synchronization protocol for wireless sensor networks, *IEEE Instrumentation and Measurement Technology Conference*, pp. 2111–21116, Warsaw, Poland, May 2007.

13. MICA Motes and Sensors, http://www.xbow.com
14. Mills, D. L., Internet time synchronization: The network time protocol, *Global States and Time in Distributed Systems,* IEEE Computer Society Press, Los Alamitos, CA, 1994.
15. Mills, D. L., Adaptive hybrid clock discipline algorithm for the network time protocol, *IEEE/ACM Transactions on Networking,* 6(5), 505–514, October 1998.
16. NCSL, Establishment and adjustment of calibration intervals, *Recommended Practice RP-1,* January 1996.
17. PalChaudhuri, S., Saha, A., and Johnson, D. B., Adaptive clock synchronization in sensor networks, *IEEE Symposium on Information Processing in Sensor Networks,* pp. 340–348, Berkeley, CA, April 2004.
18. Pottie, G. J. and Kaiser, W. J., Wireless integrated network sensors, *Communications of the ACM,* 43(5), 551–558, May 2000.
19. Su, W. and Akyildiz, I. F., Time-diffusion synchronization protocol for sensor networks, *IEEE/ACM Transactions on Networking,* 13(2), 384–397, April 2005.
20. Syed, A. and Heidemann, J., Time synchronization for high latency acoustic networks, *IEEE Infocom Conference,* pp. 892–903, Barcelona, Spain, April 2006.
21. Tay, Y. C. and Chua, K. C., A capacity analysis for the IEEE 802.11 MAC protocol, *ACM Wireless Networks Journal,* 7, 159–171, 2001.

6

Resource-Aware Localization in Sensor Networks

Frank Reichenbach
ABB Corporate Research

Jan Blumenthal
SYSGO AG

Dirk Timmermann
University of Rostock

6.1 Introduction **6**-1
6.2 Distance Estimation **6**-2
Time of Arrival • Time Difference of Arrival • Round Trip Time • Symmetric Double-Sided Two-Way Ranging • Hop Count • Lighthouse Localization System • Three/Two Neighbor Algorithm • Neighborhood Intersection • Received Signal Strength • Minimal Transmission Range
6.3 Positioning Systems and Localization Algorithms **6**-12
Positioning and Navigation Systems • Design Considerations • Algorithm Classification • Classical Methods • Proximity-Based Techniques • Optimization Methods • Iterative Methods • Pattern Matching Techniques
6.4 Conclusions **6**-24
References **6**-24

6.1 Introduction

Manually placing hundreds or thousands of tiny sensor nodes over a large area can be cost and time intensive. Thus, deploying nodes by plane or a mobile vehicle can be an adequate alternative, instead of a precise placement of each sensor node in a well-defined infrastructure. However, this implies a random distribution process, due to for instance uncontrollable natural influences like wind or flight direction of the plane. For this reason, the location of every sensor node is initially unknown. Moreover, nodes can be mobile whereas the position is time-dependent, too. The estimation of node positions is highly desirable because

- Sensor measurements without location information are useless.
- Knowledge of positions allows energy-efficient geographic routing methods without route discovery.
- Basic processes in sensor networks like self-organization or self-healing can be applied very efficiently with position information.
- Obstacles can be found and consequently bypassed easily.
- Often, the position of a sensor node itself is the demanded parameter, e.g., in applications with target tracking, where sensor nodes are attached to a moving phenomenon.

In the end, the localization process is important for network protocols. Sensor networks without position information are difficult to organize and waste valuable energy for, e.g., neighborhood discovery tasks.

Intuitively, every sensor node is equipped with an available positioning system like the global positioning system (GPS) as described in addition to in Section 6.3.1. The resource requirements of these localization systems, e.g., energy consumption, size, or price, are in contradiction to the limited resources in sensor networks. Hence, positioning systems are not applicable on every sensor node.

As an alternative, positions can be estimated relative to a randomly chosen sensor node using a Cartesian coordinate system (Section 6.3.7). Unfortunately, the localization error accumulates with an increasing distance to that sensor node. A subsequent conversion to absolute coordinates requires additional computation overhead.

The second alternative dominating the literature uses a small number of sensor nodes equipped with a positioning system. Depending on the literature these nodes are called landmarks, anchors, reference nodes, or beacons. In the following, they are defined as beacons abbreviated with B_i. On basis of absolute distances or angles to these beacons and their positions, any sensor node S_i is able to estimate its own position with an adequate localization algorithm.

Since centuries, efficient and well-defined mathematical methods have been known to estimate an unknown position with at least three beacon positions and distances to them (Section 6.2).

1. Triangulation assumes exact angle measurements. However, measuring angles is difficult in sensor nodes, due to the additional hardware overhead (Section 6.3.4).
2. More adequate in sensor networks is trilateration which requires exact distance measurements. Otherwise, imprecise distance measurements strongly degrade the precision of the trilateration. This can lead from erroneous to completely useless results and makes this technique unreliable and thus unfavorable for sensor networks.
3. Increasing the trilateration process by more input data, respectively more beacon positions with distance information, reduces the localization error significantly. Particularly, stochastic errors are decreased. Further, outliers can be detected and, thus, eliminated. As sensor networks consist of many nodes, this approach seems to be well suited. Nevertheless, an increasing amount of input data leads to a higher complexity and thus to high resource demands.

6.2 Distance Estimation

Localization requires information about the interrelation between a node and its environment. This information may comprise knowledge about neighboring nodes, connectivity metrics, positions of reference nodes, the gradient and rotation of the node, and other data. Most localization algorithms require distances between nodes and beacons in a Cartesian coordinate system.

Usually, distances cannot be measured directly. Therefore, a lot of methods have been developed to determine a distance out of other conditions. They can be divided at least into three different types of methods (Figure 6.1). In the following, the most common methods to estimate a distance are described.

6.2.1 Time of Arrival

In vacuum, a transmitted radio signal of an isotropic source propagates circularly with the speed of light at $c_{\text{Vacuum}} = 299,792\,\text{km/s}$. In matter, the speed decreases significantly due to the material properties permittivity $e = e_0 \cdot e_\text{r}$ and permeability $\mu = \mu_0 \cdot \mu_\text{r}$. In air, the speed of light equals

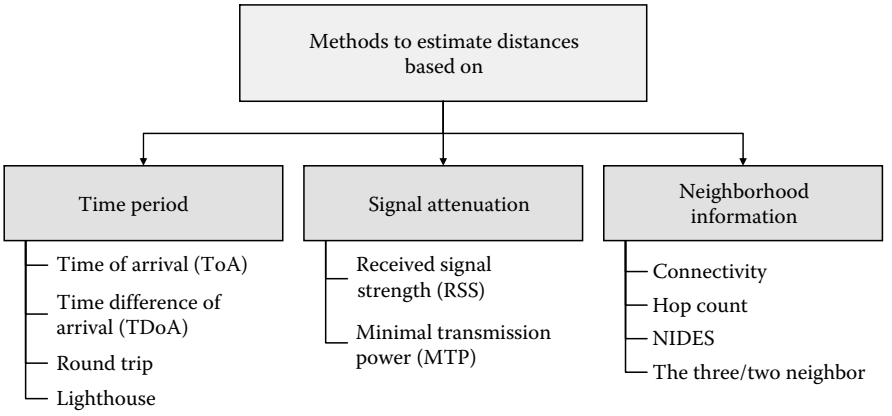


FIGURE 6.1 Classification of distance estimation methods.

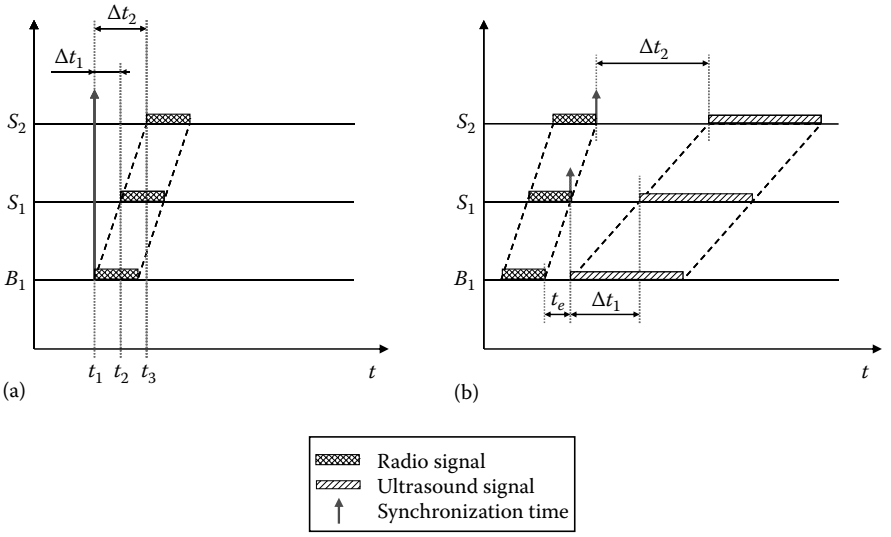


FIGURE 6.2 Distance estimation with time of arrival (ToA) at (a) global synchronization time stamp or (b) local synchronization time stamp.

$c_{\text{Air}} = 297,702\,\text{km/s}$. Hence, if material properties as well as propagation speed are known, the distance can be calculated out of the measured transmission time Δt of the signal between a sender and its receivers as well as the speed of the signal $d = c_{\text{Air}} \cdot \Delta t$ (Figure 6.2a).

The transmission time Δt is the difference between the time receiving the signal and the start time the transmission was initiated at the sender (ToA [PJ96]). Therefore, each signal contains a time stamp at which the transmission was started. After receiving the entire signal, the transmission time (time of flight, ToF) is calculated by the current time subtracted by the start time.

The calculation of the transmission time requires highly synchronized senders and receivers [Sto05]. For example, a small distance ($d = 30\,\text{cm}$) between two nodes leads in air to a very short transmission time ($\Delta t = d/c_{\text{Air}} = 0.3\,\text{m}/299,702\,\text{km/s} = 1\,\text{ns}$). Hence, the synchronization error (skew) between these two nodes must be smaller than $t_s < 1\,\text{ns}$ to detect distances around $d = 30\,\text{cm}$. Further, small distances require high time resolutions and, therefore, high clock rates to determine the transmission time very precisely.

Ignoring the sampling theorem proposed by Shannon [Sha49], the clock rate in the given example increases to at least $Clockrate = 1/t_s = 1\,\text{GHz}$. Besides the significant protocol overhead to synchronize all sensor nodes, the strong timing conditions are hard to meet in resource-aware sensor networks.

Lanzisera et al. presented [LLP06] a prototypical system to determine a distance by a reconfigurable hardware. The averaged distance error amounts to $f \approx 1\,\text{m}$. But especially at small distances ($d < 4\,\text{m}$), the distance error inceases drastically ($f \gg 1\,\text{m}$).

A method to avoid extensive synchronization between senders and receivers is the detection of time lags caused by different types of signals as described by Savvides et al. [SHS01]. Therefore, a beacon B_i transmits a radio signal with $c_1 = c_{\text{Air}}$ followed by a second signal, e.g., an ultrasound signal with a lower transmission speed (Figure 6.2b) with $c_2 = 340\,\text{m/s}$. After receiving the first radio signal, the sensor node starts a time measurement until the second signal is detected. Thus, the radio signal acts as local synchronization point at the sensor node.

Due to the slower transmission speed of the second signal, the time difference Δt between the reception of both signals increases linearly with the distance (Equation 6.1). In reality, Δt must be decreased by the ToF t_e of the radio signal which is mostly approximated to zero.

$$d = (\Delta t - t_e) \cdot c_2 \tag{6.1}$$

But unfortunately, this technique requires two transmitters at every node and is, therefore, unreasonably expensive.

6.2.2 Time Difference of Arrival

The clock and time synchronization of the whole sensor network with hundreds or thousands of sensor nodes is hard to realize. In contrast to the ToA method, the TDoA reduces the synchronization effort significantly due to synchronizing all senders only [SG99,GG03]. Synchronization of the receivers is not necessary.

To determine a distance, two senders, B_1 and B_2, transmit a radio signal at the same time (Figure 6.3a). A receiver, e.g., S_1 receives depending on the distance d_1 or d_2 one of the signals first and starts a time measurement at t_{Start}. If the second signal is received at t_{End}, the time measurement is stopped. Thus, the time difference results in $\Delta t = t_{\text{End}} - t_{\text{Start}}$.

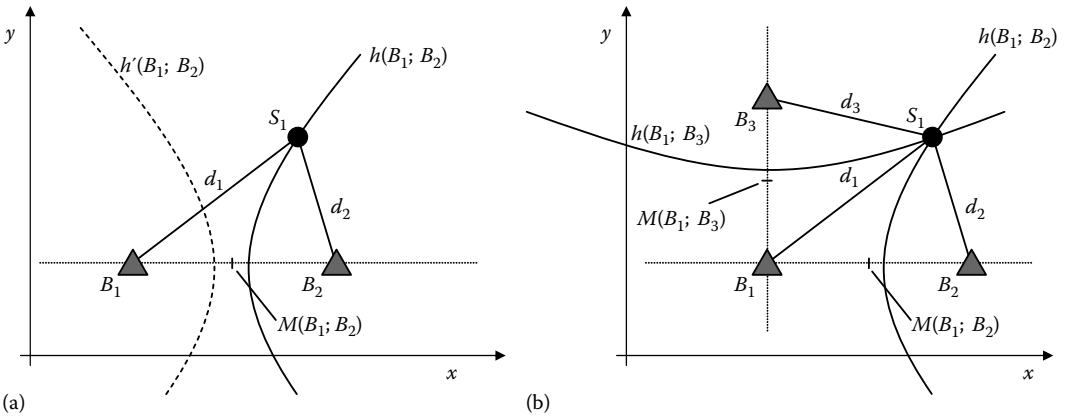


FIGURE 6.3 (a) Implicit distance estimation with time difference of arrival (TDoA) and (b) localization based on TDoA at intersection of $h(B_1; B_2)$ and $h(B_1; B_3)$.

Due to the constant signal speed in homogenous media, the difference of both distances $\Delta d = d_1 - d_2$ is proportional to the time difference Δt of the incoming signals.

$$\Delta d = \Delta t \cdot c = d_1 - d_2 \tag{6.2}$$

Dividing $d_1 - d_2$ (Equation 6.2) by Δd results in the hyperbolic function of the TDoA method.

$$1 = \frac{\sqrt{(x - x_1)^2 + (y - y_1)^2}}{\Delta d} - \frac{\sqrt{(x - x_2)^2 + (y - y_2)^2}}{\Delta d} \tag{6.3}$$

According to Equation 6.2, a sensor node is located on one of the arms of the hyperbola with the foci B_1 and B_2. The correct arm depends on the time stamp of the incoming signal. As shown in Figure 6.3a, the first incoming signal of sender B_2 implies that the distance $\overline{B_2S_1}$ is shorter than $\overline{B_1S_1}$ ($d_1 < d_2$). Therefore, S_1 can only be located on $h(B_1; B_2)$.

To detect the correct position among the hyperbola, another pair of beacons must be evaluated to define a second hyperbola $h(B_1; B_3)$. The interception of both hyperbolas, $h(B_1; B_2)$ and $h(B_1; B_3)$ defines the position of S_1 (Figure 6.3b). In some cases, both hyperbolas cut twice and deliver two possible positions. Then, a third hyperbola must be validated to prevent the ambiguities.

6.2.3 Round Trip Time

The distance estimation based on the introduced ToF methods requires highly synchronized senders or receivers and very precise clocks. Therefore, the hardware effort increases compared to other techniques. Thus, ToF methods are mostly contradictory to the design objective of developing small and resource-aware sensor networks.

A method to prevent synchronization is used by measuring the round trip time (RTT). At the beginning, a sensor node S_1 transmits a message to a target node S_2 (Figure 6.4a). This message contains the time stamp at the beginning of the transmission. The target replies to the received message immediately with an acknowledgment (ACK) after a reaction time t_R.

After reception of the ACK at S_1, the time difference of the round trip is equal to $\Delta t = \Delta t_{F12} + \Delta t_{F21} + t_{R2}$, where $\Delta t_{F12} + \Delta t_{F21}$ represents the ToF between sender and receiver. Additionally, Δt contains the reaction time of S_2 and the time Δt_{CS_i} representing the clock skew as well as the clock drift. Disregarding the usual unknown Δt_{CS_i}, a distance between S_1 and S_2 can be estimated with Equation 6.4.

$$d = c \cdot \frac{\Delta t - \Delta t_R}{2} \tag{6.4}$$

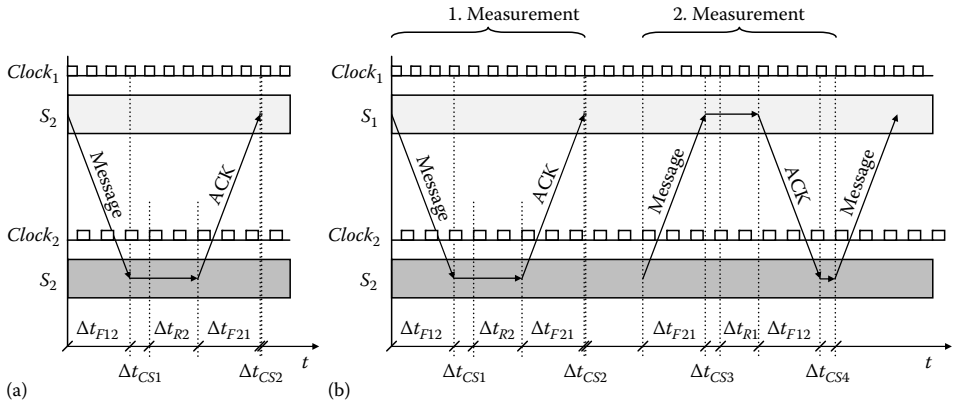


FIGURE 6.4 (a) Round trip time RTT and (b) symmetric double-sided two-way ranging SDS-TWR.

The RTT method requires a deterministic and constant reaction time to guarantee the accuracy. Therefore in most systems, the ACK is automatically generated by a specialized Media Access Control (MAC)-layer without passing higher communication layers [Sta96].

6.2.4 Symmetric Double-Sided Two-Way Ranging

The unavoidable clock drift between sender and receiver limits the accuracy of the distance estimation noticeable. To increase the accuracy, Nanotron proposed an improved method called symmetric double-sided two-way ranging (SDS-TWR) [Tec07]. Likewise in RTT, a message is sent from S_1 to S_2 and is acknowledged immediately. Based on this round trip, S_1 determines a distance $d_{12} = 0.5c \cdot (\Delta t - \Delta t_R)$. Then, node S_2 initiates a round trip by transmitting a second message to S_1. After reception of S_1's ACK, S_2 computes d_{21} and sends it to S_1. Finally, node S_1 calculates the arithmetic average of both distances d_{12} and d_{21} and, therefore, reduces clock skew effects compared to the RTT method.

6.2.5 Hop Count

In some sensor networks, the required accuracy of the determined position is very low. In these cases, also imprecise distance estimations are sufficient to approximate a position. In a multihop network with many uniformly distributed nodes, a distance can be estimated by the number of hops (hop count) a message is traveling through the network as shown in Figure 6.5. The beacons B_1 and B_2 are in line of sight of each other. Therefore, a message between both nodes must be forwarded by S_1. Thus, the estimated hop distance between B_1 and B_2 is $h_{12} = 2$ in contrast to the absolute distance $d_{e,12} = 40\,\text{m}$. Hence, this distance estimation has a scaling error of $e_{S,12} = 40/2 = 20$ and has another unit than the coordinates. If there exist several alternative routes between two nodes, the minimal hop distance is chosen.

Usually, distributed nodes are not in line with each other. Therefore, hop distances and real distances do not correlate linearly. Thus, the scaling error differs as exemplarily visualized for the connections $B_1 - B_2$ ($h_{12} = 2$, $d_{e,12} = 40\,\text{m}$, $e_{S,12} = 20$) and $B_1 - B_3$ ($h_{13} = 6$, $d_{e,13} = 100\,\text{m}$, $e_{S,13} = 16.7$). To determine a distance with correct units, Langendoen et al. proposed the Sum_Dist method [LR03,BWHF07]. In this algorithm, all single distances between beacons are simply accumulated. But besides the unhandled scaling errors, even all measurement errors are accumulated too, and finally reduce the accuracy.

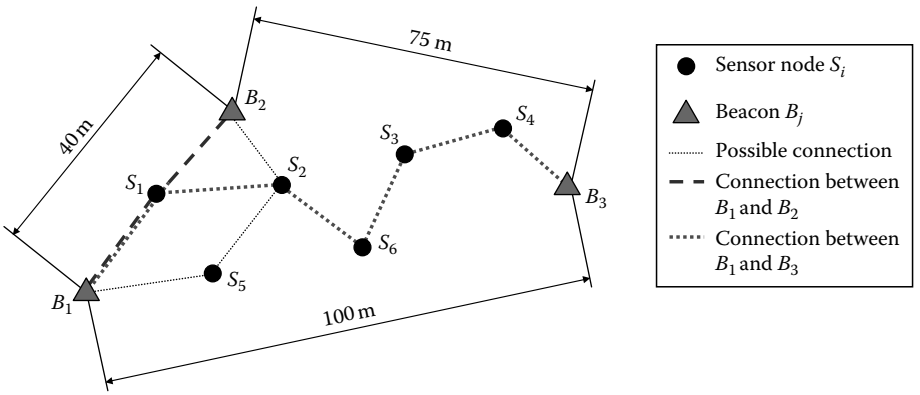


FIGURE 6.5 Hop count between several nodes.

With the presented distance vector (DV)-hop method, Niculescu and Nath decreased the impact of the scaling error and the linearization error noticeable [NN01]. They calculate an average hop size (AHS) at each beacon B_j using hop distances and real distances to all k known remote beacons B_i ($i \neq j$).

$$c_j = \frac{\sum_{i=1}^{k} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\sum_{i=1}^{k} h_i} \tag{6.5}$$

During distance estimation, the calculated AHS c_j, the actual hop count to beacon B_j, and the beacon's position are stored within the DV. A sensor node i selects the AHS c_j of the nearest beacon B_j to compute all distances d_{il} to all l remote nodes. Thus, $c_i = c_j$.

$$d_{il} = h_{il} \cdot c_i \tag{6.6}$$

The unmotivated selection of the nearest beacon and its AHS leads to a loss of potential useful information. Huang et al. therefore improved the DV-hop method by weighting all m known AHS at a sensor node i (weighted DV, WDV-hop [HS06]). According to Equation 6.7, the AHS of node S_5 is $wc_5 = (1 \cdot 17.50 + 2 \cdot 16.42 + 5 \cdot 15.90\,\text{m})/(1 + 2 + 5) = 16.23\,\text{m}$ compared to $c_5 = 17.5\,\text{m}$ of the original DV-hop method (Figure 6.5).

$$wc_i = \frac{\sum_{j=1}^{m} h_j \cdot c_j}{\sum_{j=1}^{m} h_j} \tag{6.7}$$

The distance information based on hop counts is predestinated for start values of iterative algorithms, e.g., hop terrain [SRL02] or collaborative multilateration [SPS03]. Further, DV-hop can be used to approximate positions. But in environments with many obstacles, the accuracy decreases drastically due to huge hop count produced by bypassing the holes. Another disadvantage is the decreasing connectivity to remote sensor nodes [TRV+06].

6.2.6 Lighthouse Localization System

The lighthouse localization system was introduced by Römer [Röm05]. In this system, light beams rotate around a source point with a fixed width b and constant angular speed. The time, a turn takes is defined as t_{Turn}. Due to the rotating light, a sensor node S_i detects depending on its distance to the source of light for a finite time $t_{Beam,1}$ per round (Figure 6.6a).

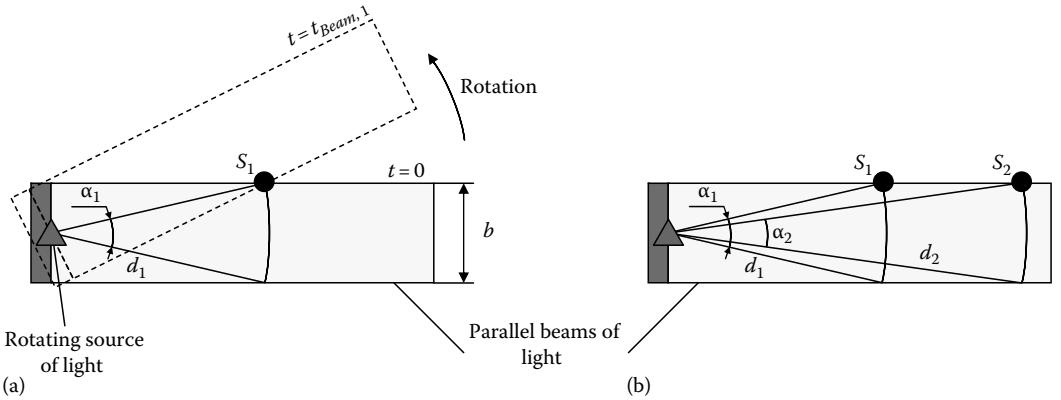


FIGURE 6.6 Lighthouse localization system: (a) Angle detection and (b) distance estimation depending on angles.

The time period to receive the rotating light depends on the distance between a node and the source point. Hence, the angle a_i decreases also with the distance ($a_2 < a_1$) as visualized in Figure 6.6b. This angle is defined as ratio of $t_{\text{Beam},i}$ to t_{Turn} and results after the conversion in radian in Equation 6.8 [SHS06].

$$\alpha = 2\pi \frac{t_{\text{Beam},i}}{t_{\text{Turn}}} \tag{6.8}$$

As in the right triangle, the sinus of the half angle $a_i/2$ is defined as ratio of the beam width b divided by two and the distance between the source of light and the sensor node.

$$\sin\left(\frac{\alpha_i}{2}\right) = \frac{b}{2d_i} \tag{6.9}$$

Inserting Equation 6.8 in Equation 6.9 and rearranging to d_i results in the final equation to determine the distance (Equation 6.10).

$$d_i = \frac{b}{2\sin\left(\pi \frac{t_{\text{Beam},i}}{t_{\text{Turn}}}\right)} \tag{6.10}$$

In reality, an uninterruptable beam of light as visualized in Figure 6.6a is hard to realize. Römer therefore proposed a realization with two single rays of light in parallel. To distinguish the first from the second ray, he marked the rays.

The lighthouse localization system is characterized by very low complexity of computation and communication, has a very high accuracy and is only slightly affected by diffraction effects. Thus in a two-dimensional area, each sensor node can determine its own position based on a trilateration provided that three distances to three different rotating beams of light are measured.

But in practice, a system of three rotating rays is hard to realize because of the high effort to install and calibrate light transceivers on all nodes. In particular, all nodes require direct line of sight to the rotating source which restricts the case of operations significantly.

6.2.7 Three/Two Neighbor Algorithm

Most of the localization algorithms require three distances d_i ($0 < i = 3$) to reference nodes to calculate a position deterministically, e.g., using a trilateration. Geometrically, the position S_1 is defined as interception of circles around these three references with the given distances.

In sensor networks, a node may know less than three distances to reference nodes. Therefore, a trilateration is plurivalent as visualized in Figure 6.7. The visualized sensor node S_2 determines its own positions at S_1 and S_2 if it knows two distances d_1 to beacon B_1 and d_2 to beacon B_2, only. Hence, S_1 and S_2 are in the neighborhood of B_1 and B_2 ($S_1 \in N(B_1) \cap N(B_2)$, $S_2 \in N(B_1) \cap N(B_2)$). Further, beacons B_1 and B_2 are in the neighborhood of S_1 and S_2 ($B_1 \in N(S_1) \cap N(S_2)$, $B_2 \in N(S_1) \cap N(S_2)$). Both beacons divide the plane into two half-planes $E|E = S_1, B_1, B_2|E \notin S_2$ and $\underline{E}|\underline{E} = S_2, B_1, B_2|\underline{E} \notin S_1$ at g_{12}.

Barbeau et al. proposed the three/two neighbor algorithm to determine a position in such configurations by determining implicit distances [BKKM04]. They assume there is a third beacon B_3 in a network (Figure 6.7b) in range of S_1 ($S_1 \in N(B_1) \cap N(B_2) \cap N(B_3)$). Thus, node S_1 knows three distances and can compute its position correctly. If B_3 is as well in range of S_2, trilateration of S_2 is unique ($S_2 \in N(B_1) \cap N(B_2) \cap N(B_3)$).

If it is not in range ($S_2 \in N(B_1) \cap N(B_2) \cap \backslash N(B_3)$) and S_2 does not know anything about B_3, the localization is plurivalent anymore. But if S_2 is able to communicate with B_1 and is therefore able to detect the correct half-plane E or $\underline{E}$ where B_3 and S_1 are placed, the localization will be unique. In

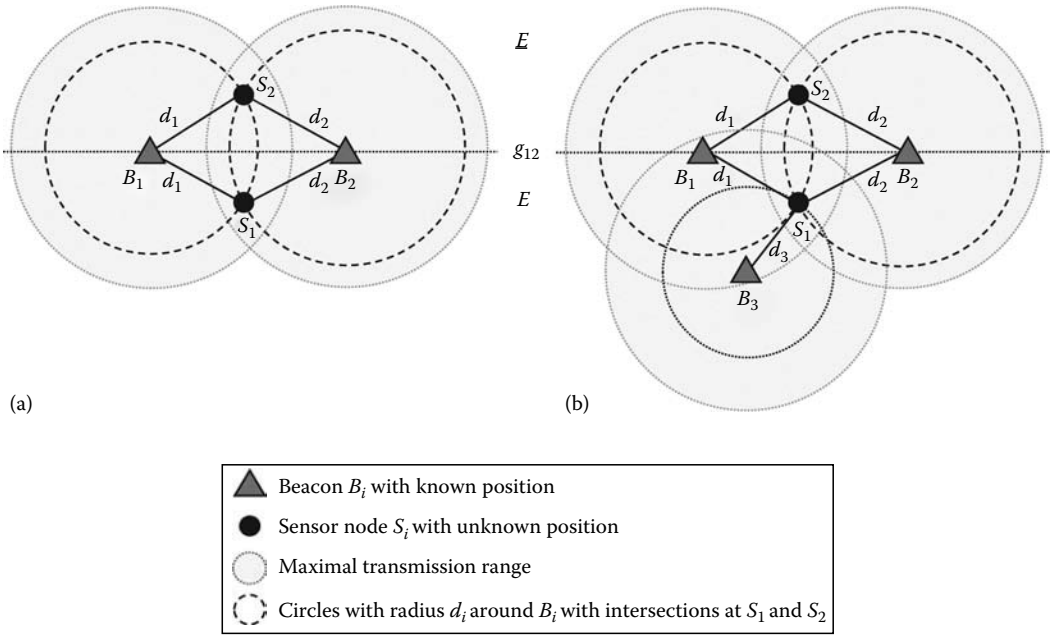


FIGURE 6.7 (a) Localization of B_1 and B_2 resulting in two positions each. (b) full localization of B_1 and B_2.

the visualized example, B_3 is located in plane E and therefore S_2 must be $\underline{E}$ ($B_3 \in N(B_1) \cap N(S_1) \rightarrow B_3, S_1 \in E \rightarrow S_2 \in \underline{E}$). In the end, this algorithm calculates a third distance implicitly, if enough neighboring information is provided.

6.2.8 Neighborhood Intersection

Buschmann et al. proposed another algorithm [BPF06] using neighboring information called neighborhood intersection distance estimation scheme (NIDES). It requires uniformly distributed sensor nodes only. A synchronization of nodes and explicit distance measurements are not necessary.

Figure 6.8 demonstrates the estimation of distance d between two sensor nodes S_1 and S_2. If both nodes are in range to each other, the transmission ranges form an overlapping region A. The

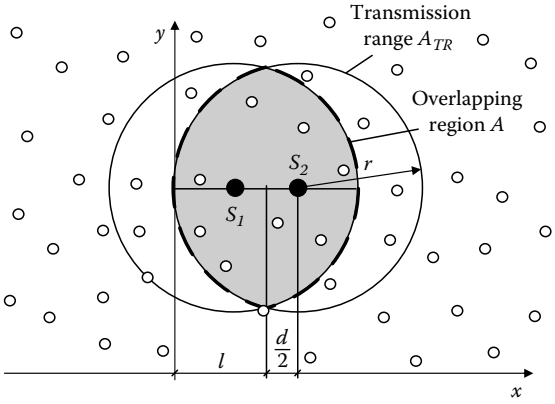


FIGURE 6.8 Neighborhood intersection distance estimation schema.

maximum width of the resulting overlapping region A is $2l$. According to the figure, the transmission range r is equal to $l + (d/2)$.

$$d = 2(r - l) \tag{6.11}$$

In an ideal environment, both sensor nodes S_1 and S_2 transmit a message. This message can be received within the radius r around the nodes. In the overlapping region A (dashed region), all n_A enclosed sensor nodes receive both messages. These sensor nodes S_i $(1 = i = n_A)$ are neighbors of both sending nodes. Due to the uniform distribution of nodes, the number of nodes $\tilde{n}$ in range of one sensor node ($A_{TR} = \pi \cdot r^2$) is constant. Since the node density μ is constant in the whole network, the following equation is formed:

$$\frac{n_A}{A} = \mu = \frac{\tilde{n}}{\pi \cdot r^2} = \frac{\tilde{n}}{A_{TR}} \tag{6.12}$$

and results in

$$A = \frac{n_A}{\tilde{n}} \cdot \pi r^2 \tag{6.13}$$

The calculation of width l can be approximated by

$$l = r + 2\sqrt{2}r \cdot \sin\left(\frac{\arcsin\left(3\sqrt{2}(A - \pi r^2)/16r^2\right)}{3}\right) \tag{6.14}$$

After calculation of l, the distance d can be determined by Equation 6.11. The achieved accuracy of NIDES oscillates between $0.25r$ and $0.1r$.

6.2.9 Received Signal Strength

In vacuum, a radio signal of a circular source propagates circularly. Generally, the transmission of radio signals is adherent with the transport of energy into the surrounding of the sender. This energy level is flatten with the distance d to the sender, but it can be detected by a receiver. In several publications, this method to detect the energy level is called received signal strength (RSS).

Friis presented an equation to compute the distance based on transmission power P_{TX}, received power P_{RX}, antenna gains (G_{TX}, G_{RX}), system losses L, and the wave length λ_0 provided ideal conditions (no reflection, no diffraction, no obstacles, etc.) are met [Fri46].

$$\frac{P_{RX}}{P_{TX}} = \frac{G_{RX}G_{TX}}{L}\left(\frac{\lambda_0}{4\pi d}\right)^2 \tag{6.15}$$

If a sensor node S_2 is able to detect the received power P_{RX} of a message, the distance between the transmitting node S_1 and S_2 can be calculated by rearranging Equation 6.15.

According to Equation 6.15, wave length λ_0 and distance d affect P_{RX} quadratically [Rap02]. The attenuation of a signal is defined as path loss (PL) which is the logarithm of transmitting power to received power in decibel.

$$\text{PL (dB)} = 10 \log \frac{P_{TX}}{P_{RX}} = -10 \log\left(\frac{G_{RX}G_{TX}}{L}\left(\frac{\lambda_0}{4\pi d}\right)^2\right) \tag{6.16}$$

To detect a signal of a transmitting node at a receiving node correctly, the receiver sensitivity $\text{PL}_{\max}$ must be higher than the PL depending on the specific distance d between both nodes. For instance,

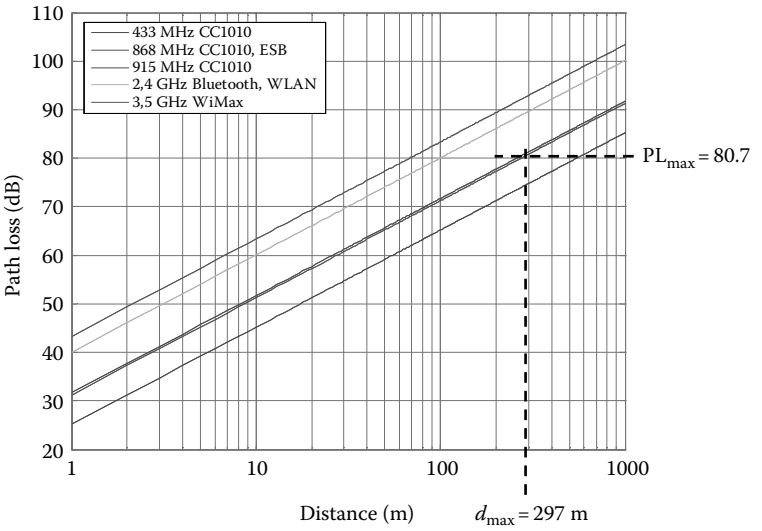


FIGURE 6.9 PL of radio signals over distance between sender and receiver at different frequencies, e.g., used by WiMax, WLAN, Chipcon CC1010, and ESB.

the receiver sensitivity of the embedded sensor boards (ESB) is $\mathrm{PL}_{\max} = 80.75\,\mathrm{dB}$ [Sch04,Dei06]. Therefore, signals can be detected in a range $0 < d < d_{\max} = 297\,\mathrm{m}$ (Figure 6.9).

$$d_{\max} = \frac{\lambda_0}{4\pi 10^{\frac{-\mathrm{PL}_{\max}}{20}}} \tag{6.17}$$

The equation of Friis (Equation 6.15) is an approximation of the received power, but it is valid only outside the Fraunhofer region $d_\mathrm{f} = 2D^2/\lambda_0$ of the sender. The Fraunhofer distance d_f depends on the maximum expansion of the antenna D ($d_\mathrm{f} >> D$) and the wave length λ_0 ($d_\mathrm{f} >> \lambda_0$).

6.2.10 Minimal Transmission Range

In systems without any analog to digital converters (A/D converter) or without circuits to convert signals, the RSS cannot be determined. In those devices, the signal strength can be measured indirectly.

In ideal environments, the antenna gains ($G_{TX} = 1$, $G_{RX} = 1$) and system losses L can be disregarded. Thus, Equation 6.16 simplifies to

$$\mathrm{PL}\,(\mathrm{dB}) = -20 \log \frac{\lambda_0}{4\pi d} \tag{6.18}$$

Usually, a sender transmits with the maximum transmission power $P_{TX,\max}$. This signal can be detected within range $d_{\max}$ by measuring the received power if it is higher or equal than $P_{RX,\min}$. The ESB for example transmits a signal at $P_{TX,\max} = 0.75\,\mathrm{mW}$. Thus, the minimal received power is $P_{RX,\min} = 0.63\,\mathrm{pW}$ at $d_{\max} = 297\,\mathrm{m}$.

$$P_{RX,\min} = \frac{P_{TX,\max}}{10^{\frac{\mathrm{PL}_{\max}}{10}}} \tag{6.19}$$

RSS-based methods determine the distance by solving Equation 6.15. Figure 6.10a visualizes the ideal relation between P_{RX} and d. Within range $0 < d < d_{\max}$, the signal can be detected and the distance can be calculated along the graph.

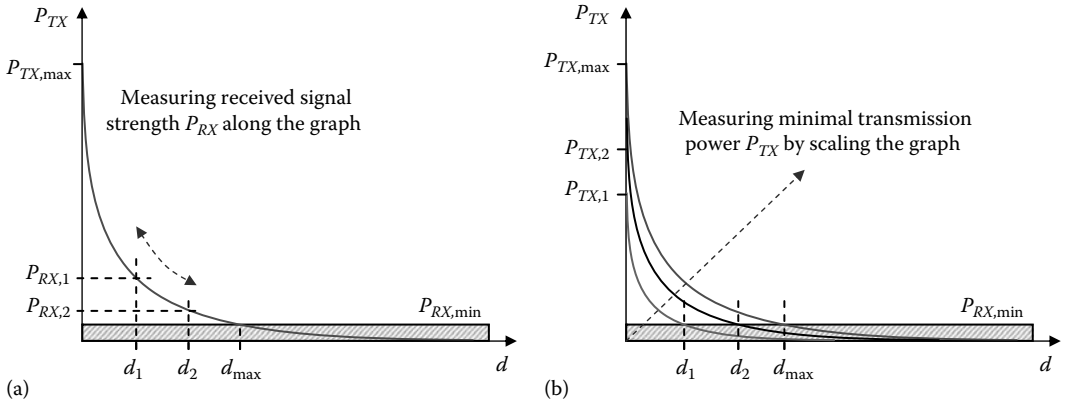


FIGURE 6.10 Distance estimation between sender and receiver with (a) RSS and (b) MTP.

A new approach to determine a distance was introduced by Blumenthal et al. in [BTB+06] called minimal transmission power (MTP). This method detects the MTP $P_{TX,\min}$ required at the sender to transmit a signal to the receiver. $P_{TX,\min}$ can be calculated by rearranging Equation 6.15 using $P_{RX,\min}$ (Equation 6.19).

In MTP, a sender continuously transmits a signal with an increasing transmission power (P_{TX} : $P_{TX,\min} < P_{TX}(R_{TX}) < P_{TX,\max}$). The transmitted signal contains the sender's adjusted transmission power P_{TX} in terms of a register value $R_{TX,i}$ to setup the transceiver. At a receiver, the signal can be detected first at $P_{RX,\min}$ depending on the distance d between both nodes. In most cases, the signal is received more than once with increasing $R_{TX,i}$. Then, only the smallest $R_{TX,i}$ is valid to determine d by solving Equation 6.20.

$$d = \sqrt{\frac{P_{TX}(R_{TX})G_{TX}G_{RX}}{P_{RX,\min}L}\left(\frac{\lambda_0}{4\pi}\right)^2} \tag{6.20}$$

In contrast to RSS where received power and distance are related along a graph with a maximum at $P_{TX,\max}$ (Figure 6.10a), MTP scales and moves the whole graph with its maximum at $P_{TX}(R_{TX})$ (Figure 6.10b).

The distance estimation using MTP is very precise, especially in very short ranges due to preventing multipath effects. Figure 6.11 shows typical results using ESB.

6.3 Positioning Systems and Localization Algorithms

This section gives an overview about both positioning systems and localization algorithms in wireless sensor networks. The most popular derivatives are described in more detail.

6.3.1 Positioning and Navigation Systems

Existing positioning systems like GPS cannot be integrated on resource limited sensor nodes due to their high energy, computation, and communication requirements. However, positioning systems are important for sensor networks where only a limited number of powerful nodes with GPS provide their position as reference.

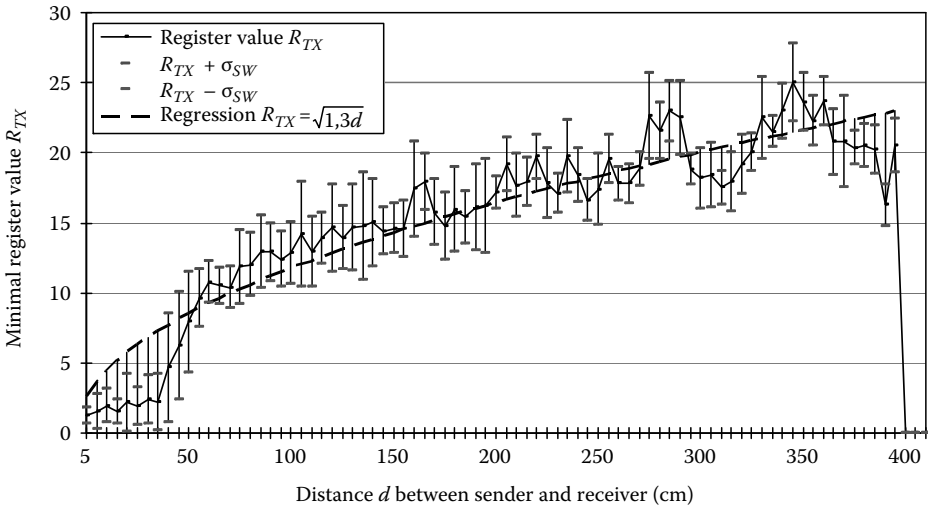


FIGURE 6.11 Register value R_{TX} of minimal transmission power $P_{TX,\min}$ depending on the distance (ESB, 40 measurements).

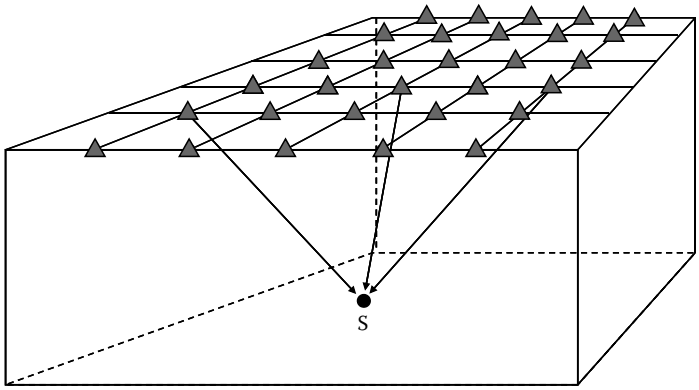


FIGURE 6.12 Schematic of the Active Bat system. (From Blumenthal, J., Resource aware and decentral localization of autonomous sensor nodes in sensor networks, PhD thesis, University of Rostock, Rostock, Germany, 2007, submitted. With permission.)

All positioning systems are based on one of the before explained distance measurement techniques (Section 6.2) or a combination of them. So far, these positioning systems are not applicable in large sensor networks, but sufficient for ad hoc networks with only some hundreds nodes.

One of the first systems was "Active Badge," which was developed by the Olivetti Research Laboratory [Wan92]. It uses badges, mounted at persons moving within a building. These badges send every 10 s an infrared signal, which allows to identify the specific person. These signals are received by sensors, installed at the ceiling of the room and then, the signals are forwarded to a central server. The server stores incoming signals with the sensor's location and thus the person's last position.

Infrared communication is only useful for short distances, because external light sources interfere the signal. For that reason Active Badge was improved by distance measurements with ultrasound. This new "Active Bat" system estimates the position of persons by lateration with distances and known sensor positions, which is demonstrated in Figure 6.12. As a result, the precision of the older Active Badge system could be improved to about 9 cm in 95% of all cases [HHS+99]. Finally, another

improvement came along with the "Cricket" system that has integrated additional hardware on the badges whereby the lateration process can be distributed [PCB00].

Another system is "RADAR," which assumes a preinstalled Wi-Fi infrastructure [BP00]. RADAR offers two methods for the localization process. The first one is using signal strength measurements and "signal to noise ratios" (SNRs) of Wi-Fi signals to estimate distances, which are basis for lateration. The second method is comparing latest measurements with stored measurements from a signal map, which was built in a preprocess. Systems that use this kind of pattern matching techniques are "MotionStar" [RBSJ79] and "Locus" [KH00].

Since 1993, the GPS has been a popular localization system used by military as well as civil applications. It achieves a precision of some centimeters. GPS works very well for outdoor environments and is limited in indoor environments, depending on the number of satellites in direct sight. Measuring the signal's ToF to geostationary satellites from a receiver is a basic requirement of GPS. With distances to satellites and their positions, GPS receiver estimate their own position via lateration [Gib96]. The standard version of GPS has been extended by several improvements like differential GPS, where fixed base stations on the ground allow error correction. Beside, the American GPS a similar system is the Russian GLONASS, which was developed for the Russian Ministry of Defense. Since years, the European Union has been working on a satellite navigation system with 30 satellites that is planed to be ready in 2013 [BCKM04].

Loran-C works similar like the before mentioned systems except that base stations are placed along the coast in a chain structure [lor07]. This system is mainly used for marine navigation.

More systems based on lateration are SpotOn [HWB00], a Bluetooth-based system developed by Feldmann [Fel03] and 3D-iD [WL98]. Other systems that must be mentioned are LANDMARC, which uses radio frequency identification (RFID) [NLLP03] or Ubisense that combines lateration and angulation with ultrawideband communication [ubi07]. For more systems, [HB01] gives a broad overview.

All discussed positioning systems are not feasible on tiny and resource constrained sensor nodes. Their energy consumption is too high and they assume a preinstalled infrastructure of base stations. Currently, only GPS is qualified, due to its global availability. The following problems are still existent:

1. *Costs*: The financial expense for a large network is not acceptable.
2. *Size*: The size of nodes equipped with GPS is very high.
3. *Energy*: GPS modules are energy intensive and require a current flow of milli amperes.
4. *Application*: GPS requires direct line of sight to at minimum four satellites. Thus, it does not work in buildings.

For these reasons, numerous research groups develop special localization algorithms that are adapted to the special sensor node's hardware and application demands.

6.3.2 Design Considerations

Resource-aware localization can only be achieved by carefully analyzing its behavior on all network layers. For example from a sensor node's view, the distributed localization on every node is privileged for robustness and reliability. But, a global network view avoids systematic errors and redundancy. Generally, the following requirements should be fulfilled:

- An estimated position must have a high precision, respectively a small localization error.
- Position information must be provided as soon as possible, immediately after network initialization, because many other protocols are based on this data.

- Financial costs for establishing a localization infrastructure must be minimal. This implies cost-efficient beacon hardware as well as minimal number of used beacons.
- The limited resources require minimal effort in computation, communication, and memory consumption.
- Localization must be processed fully distributed to achieve robustness and reliability.
- Algorithms must be adaptive to changing environmental conditions, e.g., fluctuations in the quality of the signal strength indicator.
- Mobility of nodes must be considered.

However, not all requirements can be simultaneously fulfilled. This is subject to the underlying application. For example in a highly mobile network, it is very important to estimate positions very quickly, otherwise they are not up to date. Summarized, both precise localization and minimal resource requirements are fundamentals in wireless sensor networks.

6.3.3 Algorithm Classification

Localization algorithms can be categorized as follows:

- Approximative versus exact precision
- Central versus distributed calculation
- Range-based versus distance-free (or angle)
- Relative versus absolute localization regarding point of reference
- Indoor versus outdoor usage
- Beacon-free versus beacon-based

The limits between these characteristics are floating but allow a general classification. Next, the most relevant algorithms are described in detail. Figure 6.13 gives a short comprehensive overview.

6.3.4 Classical Methods

From mathematical point of view, a position can be calculated by triangulation, namely trilateration. If angles are provided, the sensor node's position $\tilde{S}(\tilde{x};\tilde{y})$ is estimated with a triangulation as shown in Figure 6.14a. In detail, $B_1(x_1;y_1)$ and $B_2(x_2;y_2)$ are basis for a line of position $\overline{B_1B_2}$.

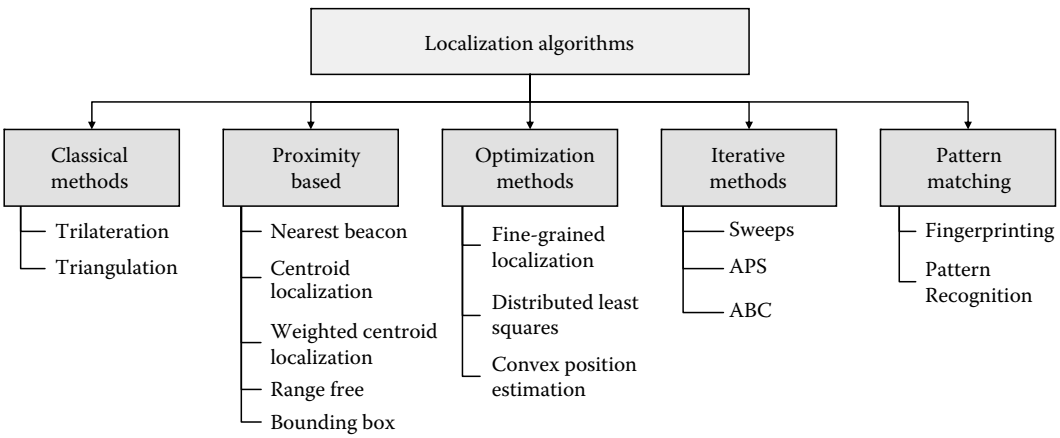


FIGURE 6.13 Classification of localization algorithms.

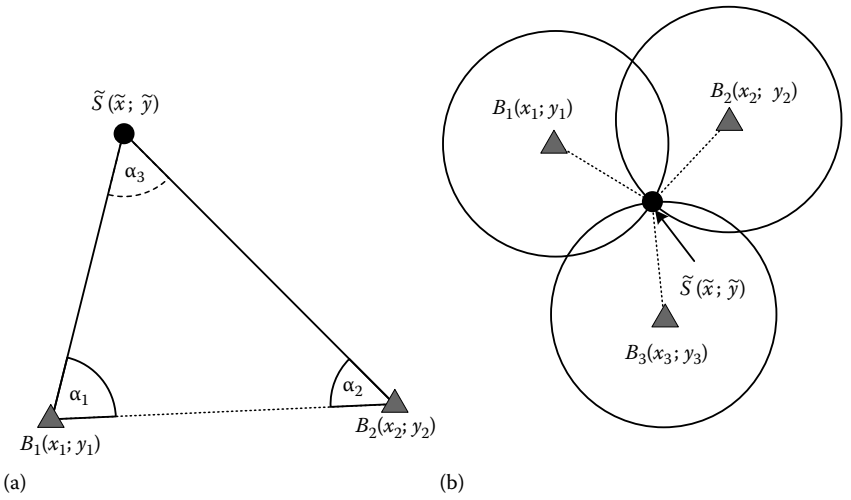


FIGURE 6.14 Localization with (a) angulation and (b) lateration. (From Reichenbach, F., Resource aware algorithms for exact localization in wireless sensor networks, PhD thesis, University of Rostock, Rostock, Germany, December 2007. With permission.)

Connecting the points $\tilde{S}(\tilde{x};\tilde{y})$, $B_1(x_1;y_1)$ and $B_2(x_2;y_2)$ leads to a triangle, where the name of this technique comes from. Moreover, measuring the angles α_1 and α_2 allows calculating the remaining angle α_3 with $\alpha_3 = 180° - (\alpha_1 + \alpha_2)$. Using the law of sine gives the estimated position with $\overline{B_1\tilde{S}} = \overline{B_1B_2}\frac{\sin(\alpha_2)}{\sin(\alpha_3)}$, respectively, $\overline{B_2\tilde{S}} = \overline{B_1B_2}\frac{\sin(\alpha_1)}{\sin(\alpha_3)}$.

Assuming that, instead of angles, distances are known between nodes, a position can be determined via trilateration (Figure 6.14b). A minimum of three distances $\overline{B_1\tilde{S}}$, $\overline{B_2\tilde{S}}$, $\overline{B_3\tilde{S}}$ and three beacon positions build a system of equations:

$$\left(\overline{B_i\tilde{S}}\right)^2 = \left(B_i(x_i) - \tilde{S}(\tilde{x})\right)^2 + \left(B_i(y_i) - \tilde{S}(\tilde{y})\right)^2 \tag{6.21}$$

6.3.5 Proximity-Based Techniques

6.3.5.1 Nearest Beacon

Simply choosing the nearest beacon position, for example, by determining the highest received signal strength indication (RSSI), is a possibility to solve the localization problem. Although, this can be achieved with minimal computation effort, the localization error may be very high. Nevertheless, the estimated position can be used as start value for iterative methods that increase the precision [BRBT06].

6.3.5.2 Coarse-Grained Localization

Centroid localization (CL) was first published by Bulusu as "GPS-less low cost outdoor localization for very small devices" in [Bul00]. The algorithm completely avoids explicit distance measurements, but assumes a grid-based beacon placement with constant distances q between each other as demonstrated in Figure 6.15. To localize nodes, b beacons $B_1(x_1;y_1)\cdots B_b(x_b;y_b)$ are aligned in a grid. The transmission range of beacons is assumed to be circular.

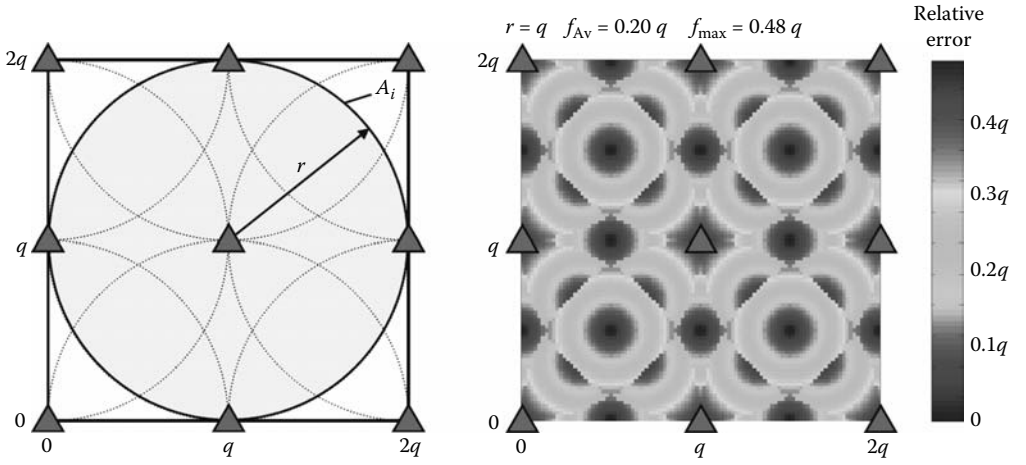


FIGURE 6.15 Scenario with a 3 × 3 grid of beacons: (a) resulting overlaps and (b) localization error. (From Blumenthal, J. and Timmermann, D., *4th IEEE/ACM International Conference on Distributed Computing in Sensor Systems (DCOSS 2008)*, pp. 47–48, Santorini, Griechenland, June 2008. With permission.)

CL starts on every beacon by sending a packet including its own position and network address. Sensor nodes within the beacon's transmission range receive corresponding packets and save its content. At the end of this phase, every sensor node received n packets and determines the position with the centroid formula:

$$\tilde{S}(\tilde{x};\tilde{y}) = \frac{1}{n}\sum_{j=1}^{n} B_j(x_j;y_j) \tag{6.22}$$

If no beacon position is received, a sensor node's position cannot be estimated. Analysis in an outdoor testbed with a quadratic sensor field of side length a = 10 m, four beacons at the corners and 121 test points resulted in an averaged localization error of f_{Av} = 1.83 m with standard deviation $\sigma_{f_{Av}}$ = 1.07 m [Bul00]. Additional analysis showed that the error strongly depends on the beacons transmission range and placement [BRHT04,Rei04]. For that, [RBT06] presents a simple equation to determine the optimal transmission range in terms of a specific grid width. Moreover, Bulusu et al. and Salomon suggest strategies for a better beacon placement, which also reduces the localization error significantly [BHE01,SB05]. In finite sensor networks with high transmission ranges, the CL is characterized by a very high localization error near the borderlines of the network. This error can be reduced significantly as proposed by Blumenthal [Blu].

6.3.5.3 Weighted Centroid Localization

Weighted CL (WCL) is an extension of CL by including distances in form of weights in the centroid formula. WCL was first published in [BRT05] and features randomly distributed beacons opposite to the grid alignment CL assumes.

Figure 6.16 shows a sensor node, which was localized by CL and also WCL. Four beacons are placed at the corners of the sensor field. Whereas CL leads to a high error, WCL reduces the error, because nearer beacons pull the sensor node's position stronger to their own position than to farer beacons. This also works in case of noisy distance measurements. Beacons have constant and ideal concentric transmission ranges. In a first phase, beacons send their position and network address to all sensor nodes in range. These nodes save this information. Contrary to CL, the distances to all beacon must be measured by one of the techniques described in Section 6.2. After a defined timeout, phase 2 starts.

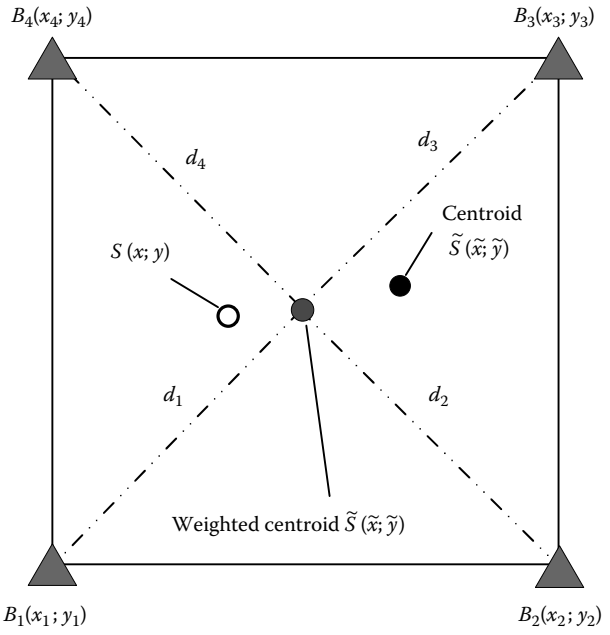


FIGURE 6.16 WCL compared to CL. (From Reichenbach, F., Resource aware algorithms for exact localization in wireless sensor networks, PhD thesis, University of Rostock, Rostock, Germany, December 2007. With permission.)

This implies calculating the position at every sensor node by a weighted centroid determination:

$$\tilde{S}_i(\tilde{x}_i; \tilde{y}_i) = \frac{\sum_{j=1}^{n} \left(w_{ij} \cdot B_j\right)}{\sum_{j=1}^{n} w_{ij}} \tag{6.23}$$

In Equation 6.23, the parameter w_{ij} represents the weight between sensor node i and beacon j. To achieve the above-mentioned characteristics, the weight can be calculated by the inverse of the distance.

$$w_{ij}\left(d_{ij}\right) = \frac{1}{d_{ij}^{g}} \tag{6.24}$$

In Equation 6.24, a new parameter was introduced—the degree g, which amplifies shorter distances to beacons.

Simulation results illustrated in Figure 6.17 show the localization error with several minima depending on the ratio between transmission range and degree. In terms of minimal energy consumption, all sensor nodes can estimate a position if r is greater than $r_{\min} = 0.71q$ using a weight function $w = 1/d$. If $r < r_{\min} = 0.71q$ (critical area) not all sensor nodes are able to estimate a position, because not enough beacons are in range. However, if high precision instead of minimal energy consumption is demanded, then the optimum exists at $r_{\text{opt}} = 0.95q$, because there, the smallest localization error is achieved. Although a degree of $g = 1$ yields in the best results, it is highly fluctuating and thus critical to apply. Higher degrees are more stable, which may be better in practice.

In conclusion, WCL is very resource aware, because every beacon sends only one short packet. Further on, a sensor node must receive n packets, measures the distance and executes a calculation with a time complexity of $\mathcal{O}(6n - 2)$.

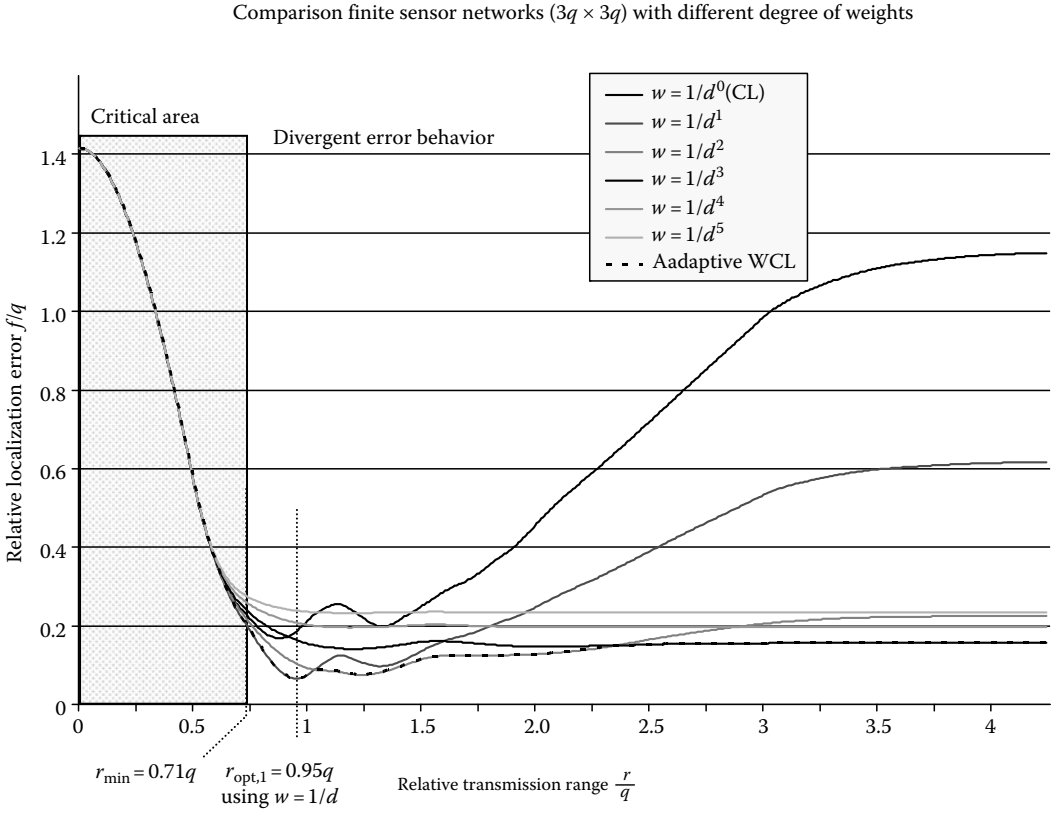


FIGURE 6.17 Localization error using different degrees. (From Blumenthal, J., Grosmann, R., Golatowski, F., and Timmermann, D., *1st European ZigBee Developer's Conference* (*EuZDC*), Munich, Germany, June 2007. With permission; Blumenthal, J., *IEEE International Symposium on Intelligent Signal Processing, WISP 2007*, Madrid, Spain, October 2007. With permission.)

6.3.5.4 Range-Free Localization

The range-free localization algorithm is also known as "approximate point in triangulation" (APIT) and bases on triangular surfaces. The algorithm was first published in [HHB+03]. He et al. reduce the influence of absolute distance measurements, due to the high error that can be expected.

Likewise in CL, every beacon transmits its position in the sensor field. By permuting all beacon positions $b!/3! \cdot (b - 3!)$, every sensor node determines all resulting triangles. Then, each triangle is checked by the "point in triangulation" (PIT) test. This check allows to make a decision if the sensor node is placed on the triangle surface or not. After this test is finished, sensor nodes know all triangles on which they are placed. All beacons constructing these triangles are used to estimate the sensor nodes position by a centroid calculation with their positions. This is shown in Figure 6.18a.

The centerpiece of APIT is the PIT test. Theoretically, a sensor node is outside a triangle of three beacon positions if the following assumption is true: A mobile sensor node is slightly shifted in any direction Δd, then all three distances to these three beacons must increase or decrease simultaneously. Otherwise, the node must be inside a triangular (Figure 6.18b).

He et al. take advantage of a large node density in sensor networks. Figure 6.18c exemplary illustrates that a sensor node may have connection to four very close neighbors. These neighbors measure the RSSI to all three required beacons and send this information to the sensor node under test. This

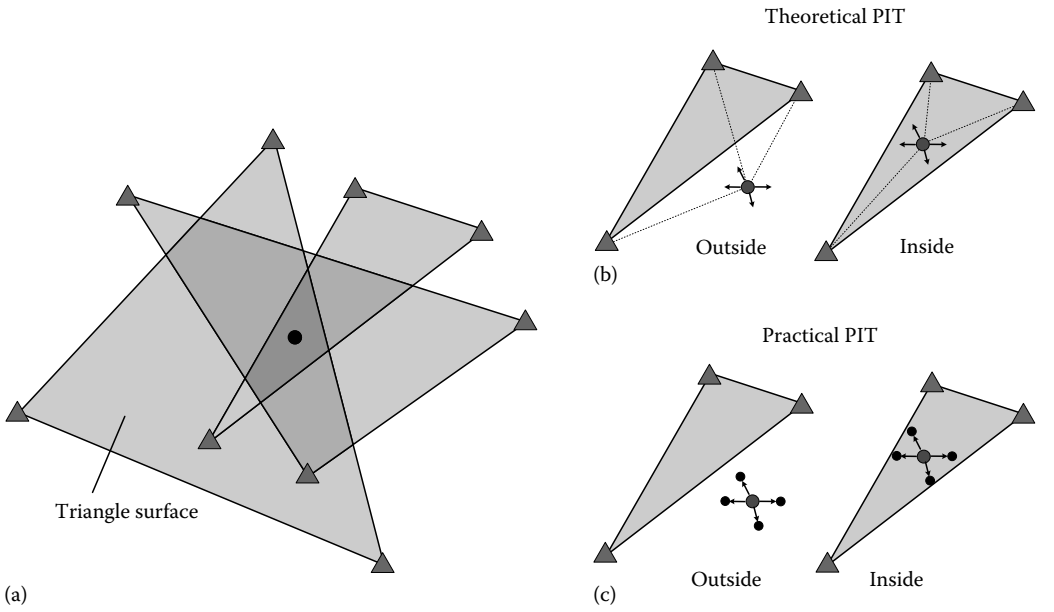


FIGURE 6.18 (a) Overlapping of triangles are basis of APIT, (b) theoretical PIT test with distances and a moveable sensor node, and (c) practical PIT test with neighbors and RSSI values. (From Reichenbach, F., Born, A., Bill, R., and Timmermann, D., Lokalisierung in Ad Hoc Geosensornetzwerken mittels geodatischer Ausgleichungstechnik, *GIS: Zeitschrift für Geo Informationssysteme*, ABC Verlag GmbH, Heidelberg, Germany, 2008, pp. 4–16. With permission; Reichenbach, F., Resource aware algorithms for exact localization in wireless sensor networks, PhD thesis, University of Rostock, Rostock, Germany, December 2007. With permission.)

node checks if all four neighboring RSSI are higher or lower simultaneously. This is similar to the theoretical PIT test and avoids absolute distances as well as moving capabilities.

6.3.5.5 Bounding Box Algorithm

The bounding box algorithm was published in [SS02] and is also based upon beacons. In this algorithm, beacons transmit their position with a specific transmission power, respectively within a well-defined transmission range r. Next, sensor nodes receive beacon packets and determine distances to beacons by one of the measurement techniques described in Section 6.2. Then, each sensor node calculates a quadratic surface $A_i = (2d_{ij})^2$ including every $B_i(x_i; y_i)$, where $B_i(x_i; y_i)$ is the middle of the square (Figure 6.19a). By comparing the minimal and maximal coordinates (Equation 6.25) of all resulting squares A_i, sensor nodes narrow down the surface on which they are placed. This remaining square A_S is called bounding box (Figure 6.19b). These rough position estimates can later be used for iterative approaches.

$$\begin{aligned} x_{\min} &= \max(x_i - d_{ij}) \le x_j \le \min(x_i - d_{ij}) = x_{\max} \\ y_{\min} &= \max(y_i - d_{ij}) \le y_j \le \min(y_i - d_{ij}) = y_{\max} \end{aligned} \tag{6.25}$$

A disadvantage of the bounding box algorithm is that a small localization error can only be achieved if S_j is placed within a convex, by beacons limited, hull A_S. Otherwise, the bounding box, and therefore the localization error, increases as shown in Figure 6.19c. Moreover, the algorithm behaves instable if distance measurements are very noisy, because probably overlapping surfaces are missed.

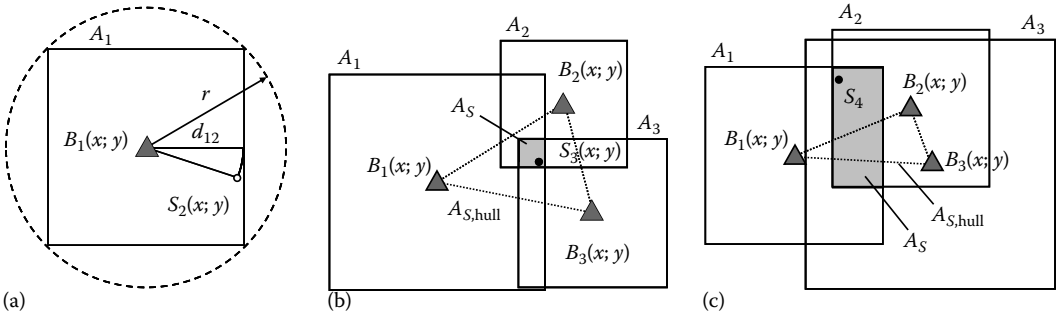


FIGURE 6.19 (a) Bounding box of $B_1(x_1; y_1)$, (b) bounding box A_S of sensor node $S_3(x_3; y_3)$ ($S_3 \in A_{S,hull}$), and (c) bounding box A_S ($S_4 \notin A_{S,hull}$). (From Blumenthal, J., Resource aware and decentral localization of autonomous sensor nodes in sensor networks, PhD thesis, University of Rostock, Rostock, Germany, 2007, submitted. With permission.)

6.3.6 Optimization Methods

6.3.6.1 Fine-Grained Localization

One of the most cited precise localization algorithms in literature is probably the "fine-grained localization" (FGL) by Savvides et al. [SHS01]. The name origins by achieving a fine-grained resolution of position estimations. FGL demands minimal three beacon positions and can be executed on every sensor node, respectively completely distributed. The core of FGL is the least squares (LS) method. With more beacon positions than needed, the precision of the algorithm significantly increases. Beside the basic atomic multilateration, the iterative and the collaborative multilateration were presented (Figure 6.20).

In atomic multilateration, every sensor node measures distances to all beacons in range. Both the Euclidean distances and the beacon's positions lead to the equation for the distance error:

$$f_i(\tilde{x}_i, \tilde{y}_i, c) = c\Delta t_{ij} - \sqrt{(x_j - \tilde{x}_i)^2 + (y_j - \tilde{y}_i)^2} \tag{6.26}$$

In Equation 6.26, $c\Delta t_{12}$ represents a distance which may be calculated by measuring the time difference Δt_{12} multiplied with the propagation speed c of the signal. Savvides presented specialized sensor nodes (Medusa) transmitting ultrasound signals with a propagation speed of c to determine the time Δt_{12} a signal needs from sensor node i to beacon j. Rearranging, squaring, and linearizing of Equation 6.26 forms Equation 6.27.

$$-x_j^2 - y_j^2 = \tilde{x}_i^2 + \tilde{y}_i^2 + \tilde{x}_i(-2x_j) + \tilde{y}_i(-2y_j) - c^2\Delta t_{ij}^2 \tag{6.27}$$

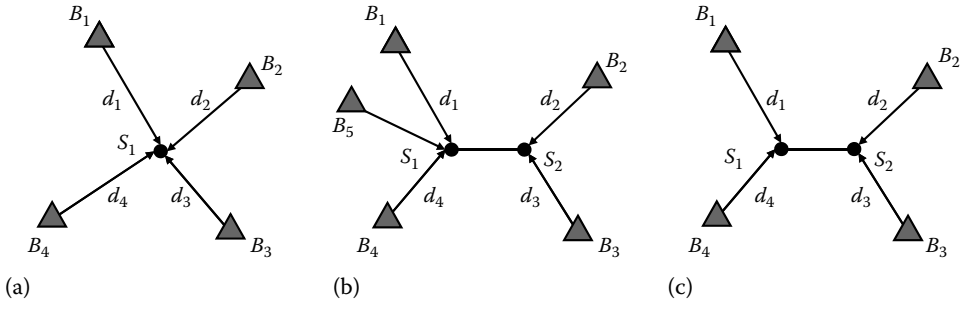


FIGURE 6.20 (a) Atomic, (b) iterative, and (c) collaborative multilateration.

A set of three or more of these equations builds a system of equations ($1 \leq j \leq h$). Subtracting the last equation h from all other equations decrements the number of equations and results in

$$-x_j^2 - y_j^2 + \tilde{x}_h^2 + \tilde{y}_h^2 = 2x_i(\tilde{x}_h - x_j) + 2\tilde{y}_i(\tilde{y}_h - y_j) + c^2(t_{jh}^2 - t_{ij}^2) \tag{6.28}$$

Equation 6.28 can be represented by the matrix form $A\mathbf{x} = \mathbf{b}$ and is solved by a minimum LS estimator $\mathbf{x} = (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}\mathbf{b}$. An extension of the basic algorithm is the iterative multilateration (Figure 6.20b), where every sensor node with an estimate starts to act as a beacon. That means, every sensor node requests positions from its direct neighbors, which are then included in a refinement process. This method strongly depends on the precision of the first estimates.

If node density is low and some nodes cannot reach at least three beacons, normally the multilateration fails. For this case, collaborative multilateration was proposed, which extends the one hop lateration to multiple hops by considering normal sensor nodes (Figure 6.20c).

6.3.6.2 Distributed Least Squares

In the following, we explain the distributed least squares (DLS) algorithm. DLS features high precision at minimal power consumption at sensor nodes and was developed by Reichenbach et al. [RBTB06,Rei07]. DLS starts with the creation of an over determined system of non linear Euclidean distances of the form $(\tilde{x} - x_i)^2 + (\tilde{y} - y_i)^2 = r_i^2$ with $i = 1, 2, \ldots, m$ (where m is the number of beacons, $\tilde{S}(\tilde{x}; \tilde{y})$ is the required position, $B_i(x_i; y_i)$ is the position of beacon i and r_i is the distance between them). This system of equations is linearized with the method described in [MH95]. This leads to the form $A\mathbf{x} = \mathbf{b}$, where A is the coefficient matrix, $\mathbf{b}$ is the right side vector, and $\mathbf{x}$ is the solution vector. By applying the LS method, we obtain the normal equation $\mathbf{x} = \left(A^{\mathrm{T}}A\right)^{-1}A^{\mathrm{T}}\mathbf{b}$.

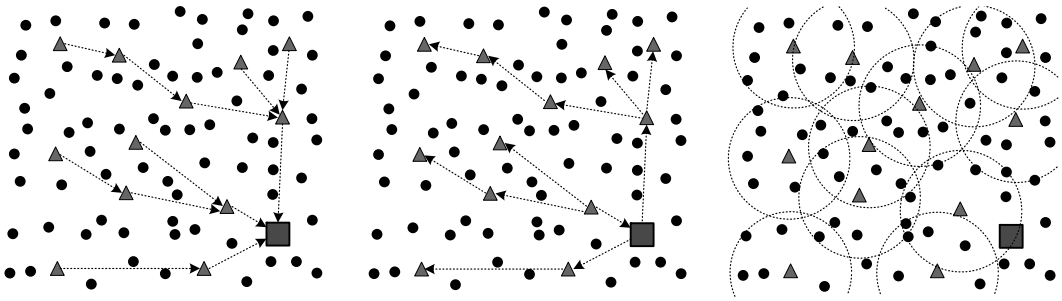
The matrices in this equation have two important advantages. First, all elements in the coefficient matrix A are generated by beacon positions $B_1(x_1; y_1) \cdots B_m(x_m; y_m)$ only. By assuming that, we can establish communication links between all sensor nodes and all beacons (e.g., by multihop techniques), then matrix A is the same on every sensor node. Second, the vector $\mathbf{b}$ contains the distances between sensor nodes and beacons $r_1 \cdots r_m$ that must be estimated at every sensor node independently. Given these facts, the normal equation is split into two parts—a more complex part, the precalculation: $A_{\mathrm{p}} = \left(A^{\mathrm{T}} \cdot A\right)^{-1} \cdot A^{\mathrm{T}}$ and a simple part: $A_{\mathrm{p}} \cdot \mathbf{b}$, which is further called the postcalculation.

The precalculation is executed on a powerful base station, which additionally avoids high redundancy, because this precalculation would normally be executed on all sensor nodes separately. But it is very important to emphasize that the precalculation is identical on every sensor node. Thus, it is calculated only once, conserving expensive energy resources. The simple postcalculation is executed at every sensor node with its individual distance measurements to all beacons. This approach complies with two important design strategies for algorithms in large sensor networks—a resource-aware and distributed localization procedure. Finally, this can be achieved with less communication overhead required for other exact algorithms.

At this point, we briefly describe the algorithm process. DLS is divided into three phases, which are shown in Figure 6.21. In phase 1, all beacons send their position $B_i(x_i; y_i)$ hop-by-hop over their beacon neighbors to the base station (Figure 6.21a). In phase 2, the base station starts generating the initial matrices and computes A_{p} (Figure 6.21b). The result is sent over beacons to all sensor nodes, which in phase 3 estimate their position after measuring the distances to all beacons and executing the postcalculation (Figure 6.21c).

6.3.6.3 Convex Position Estimation

The "convex position estimation" (CPE) of Doherty et al. estimates a position by using geometric constraints between nodes [DPG01]. In more detail, a sensor network consists of nodes (positions) and edges (communication links). The existence of some beacons is also assumed. Radial and angular



(a) Phase 1 (b) Phase 2 (c) Phase 3

FIGURE 6.21 Phases of DLS algorithm. (From Reichenbach, F., Resource aware algorithms for exact localization in wireless sensor networks, PhD thesis, University of Rostock, Rostock, Germany, December 2007. With permission.)

relations between two communicating nodes can be transfered into a system of equations. Radial relations are given by an ideal circular transmission range. For example, if two nodes with $r = 10\,\text{m}$ establish a symmetric link, then both nodes are in range between 0 and 10 m. By adjustable transmission ranges more constraints can be considered.

If angles are provided, angular relations can be found. In [DPG01], optical techniques for angle measurements are suggested. Regardless, which relation was used, all relations are sent to a central node with sufficient resources. This high powered node estimates the node's position in a central convex optimization process and send every position back to the specific node. This convex optimization is very efficient and therefore very precise for large sensor networks with a high node density.

6.3.7 Iterative Methods

6.3.7.1 Sweeps

In [GBY+06], Goldenberg et al. present an approach with so-called "Sweeps." This algorithm is processed iteratively, whereby sensor nodes calculate all their possible positions on basis of a trilateration. As many positions as beacon triples exist are possible to determine. The "Sweeps" algorithm differs in (1) nodes with exact one position estimate, (2) nodes with more than one position estimate, and (3) nodes without a position. The third case occurs if less then three beacons are in range of a node. Sweeps is arranged in different phases. First, only beacons provide a position to sensor nodes. Next, in phase 2, sensor nodes collect information of neighboring sensor nodes and calculate all possible trilaterations, respectively all estimated positions. At the end, one position can be determined.

6.3.7.2 Ad Hoc Positioning System

In the "ad hoc positioning system" initially beacons flood their positions into the network [NN01, NN03]. On every sensor node, first minimum three distances are determined by a DV-hop technique (Section 6.2), which are bases for a trilateration process. In [SRL02], a refinement phase is appended to the algorithm, in which initial position estimates are send to neighbors. This information is included in a second estimation that results in better precision.

6.3.7.3 Assumption-Based Coordinates

The "assumption-based coordinates" (ABC) algorithm starts the localization process with an initial sensor node S_0, whose position is placed in the origin of a relative coordinate system $\tilde{S}_0(0;0)$ [SRB01, SRL02]. In a next step, two neighboring nodes of S_0 determine their distances to S_0, which are d_{01} as well as d_{02} and the distance between each other d_{12}. This allows estimating a first position relative to

$\tilde{S}_0$, which is $\tilde{S}_1(d_{01};0)$. Assuming that the square root of $\tilde{y}_2$ is positive, the second node's position is given to $\tilde{S}_2\left(\frac{d_{01}^2+d_{02}^2+d_{12}^2}{2d_{01}};\sqrt{d_{02}^2-\tilde{S}_2\left(\tilde{x}_2^2\right)}\right)$. The resulting system of equations can be easily solved. All remaining sensor nodes determine their own position estimate iteratively by trilateration until the whole network is covered. If later absolute coordinates are known, then these relative coordinates can be transformed in an absolute coordinate system, which requires additional computation overhead. Similar algorithms are suggested in [CHH02,PBDT03].

6.3.8 Pattern Matching Techniques

The propagation character of electromagnetic waves in a static environment with constant beacon positions is subject to a specific signature. This effect is basis for pattern matching techniques, whereas signal characteristics are measured at many points in the room. This data is stored in a so-called signal map. Signal maps, for example, consist of many RSSI at different coordinates. Every sensor node initially saves one signal map, which is after deployment basis for a comparison algorithm. In detail, sensor nodes measure RSSI at the location, where they were deployed. Then, measured values are compared with values in the signal map, whereas the best and thus most probable match gives a position estimate. This method is also known as "pattern recognition" or "fingerprinting" and developed by Bahl and Padmanabhan [BP00].

6.4 Conclusions

Due to the strong limitations in sensor networks (e.g., very small form factor, limited capacity of energy), an efficient localization method requires small communication overhead and energy-aware algorithms to meet the conditions. In practice, a precise localization is impeded additionally by faulty input values caused by measurements of the environment. Supplementally, underlying ideal models lead in reality to distorted data as can be seen by the distance estimation based on circular signal attenuation.

The localization error generally fluctuates depending on the specifics of the selected localization algorithm. But, the precision of the localization is significantly affected by the calibration of the system, e.g., the adjusted transmission power of the transceiver or the ratio between beacons and sensor nodes.

Faulty or heavily unsteady input data lead inevitably to an increasing localization error. Especially, classical localization algorithms (trilateration, triangulation) or pattern matching are very vulnerable whereas proximity-based algorithms react more stably. In contrast, iterative methods and optimization methods behave more robust in systems with a high node density or with lots of measured data. Due to the redundant data, sensor nodes estimate positions optimally by using more information than needed. However, the resource requirements increase noticably in these networks. Thus, choosing a localization algorithm usually depends on a lot of constraints imposed by the specific application.

References

[BCKM04] Bill, R., Cap, C., Kofahl, M., and Mundt, T.: Indoor and outdoor positioning in mobile environments, a review and some investigations on WLAN positioning. *Geographic Information Sciences* 10 (2004), 91–98.

[BHE01] Bulusu, N., Heidemann, J., and Estrin, D.: Adaptive beacon placement. In: *21st International Conference on Distributed Computing Systems*, 2001, IEEE Computer Society, Washington, DC, pp. 489–498.

[BKKM04] Barbeau, M., Kranakis, E., Krizanc, D., and Morin, P.: Improving distance-based geographic location techniques in sensor networks. In: *3rd International Conference of Ad-Hoc Networks & Wireless*, 2004, Vancouver, Canada.

[Blu] Blumenthal, J.: *Resource Aware and Decentral Localization of Autonomous Sensor Nodes in Sensor Networks*, PhD Thesis, University of Rostock, 2008.

[BP00] Bahl, P. and Padmanabhan, V.N.: RADAR: An in-building RF-based user location and tracking system. In: *IEEE Infocom*, 2000, Tel-Aviv, Israel, pp. 775–784.

[BPF06] Buschmann, C., Pfisterer, D., and Fischer, S.: Estimating distances using neighborhood intersection, *IEEE Conference on ETFA*, 2006, Prague, Czech Republic.

[BRBT06] Born, A., Reichenbach, F., Bill, R., and Timmermann, D.: Bestimmung Optimaler Startwerte zur Exakten Lokalisierung mittels Geodätischer Ausgleichung. In: *5. GI/ ITG KuVS-Fachgespräch Drahtlose Sensornetzwerke, Technischer Bericht 2006/7*, 2006, Stuttgart, Germany, pp. 93–98.

[BRHT04] Blumenthal, J., Reichenbach, F., Handy, M., and Timmermann, D.: Low power optimization of the coarse grained localization algorithm in wireless sensor networks. In: *1st Workshop on Positioning, Navigation and Communication*, Shaker Verlag, March 2004, Hannover, Germany, pp. 137–146.

[BRT05] Blumenthal, J., Reichenbach, F., and Timmermann, D.: Position estimation in ad hoc wireless sensor networks with low complexity. In: *Joint 2nd Workshop on Positioning, Navigation and Communication 2005 and 1st Ultra-Wideband Expert Talk*, Shaker Verlag, März 2005, Hannover, Germany, pp. 41–49.

[BTB+06] Blumenthal, J., Timmermann, D., Buschmann, C., Fischer, S., Koberstein, J., and Luttenberger, N.: Minimal transmission power as distance estimation for precise localization in sensor networks, *International Wireless Communications and Mobile Computing Conference*, 2006, Vancouver, Canada, pp. 1331–1336.

[Bul00] Bulusu, N.: GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine* 7 (2000), 28–34.

[BWHF07] Buschmann, C., Werner, C., Hellbrück, H., and Fischer, S.: NIDES: Ein Verfahren zur Multihop-Distanzschätzung mittels Nachbarschaftsanalyse. In: *KiVS*, 2007, pp. 151–162.

[CHH02] Capkun, S., Hamdi, M., and Hubaux, J.-P.: GPS-free positioning in mobile ad hoc networks. *Cluster Computing* 5 (2002), 157–167.

[Dei06] Deinert, F.: Mathematische Modelle zur Wellenausbreitung für die Simulation drahtloser Netze. In: *Seminar Technische Informatik*, 2006, FU Berlin, Germany.

[DPG01] Doherty, L., Pister, K.S.J., and Ghaoui, L.: Convex position estimation in wireless sensor networks. In: *20th Annual Joint Conference of the IEEE Computer and Communications*, 2001, Anchorage, AK, pp. 1655–1663.

[Fel03] Feldmann, S.: An indoor Bluetooth-based positioning system: Concept, implementation and experimental evaluation. In: *2nd International Workshop on Wireless Ad Hoc Networking*, 2003, Las Vegas, NV, pp. 23–26.

[Fri46] Friis, H.T.: A note on a simple transmission formula. *Proceedings of the IRE* 34 (1946), 254–256.

[GBY+06] Goldenberg, D.K., Bihler, P., Yang, Y.R., Cao, M., Fang, J., Morse, A.S., and Anderson, B.D.O.: Localization in sparse networks using sweeps. In: *12th Annual International Conference on Mobile Computing and Networking*, 2006, ACM Press, Los Angeles, CA, pp. 110–121.

[GG03] Gustafsson, F. and Gunnarsson, F.: Positioning using time-difference of arrival measurements. In: *IEEE International Conference on ICASSP*, 2003, Hong Kong, PRC, pp. 553–556.

[Gib96] Gibson, J.: *The Mobile Communications Handbook*. CRC Press, Boca Raton, FL, 1996.

[HB01] Hightower, J. and Borriella, G.: Location systems for ubiquitous computing. *IEEE Computer* 34 (2001), 57–66.

[HHB+03] He, T., Huang, C., Blum, B.M., Stankovic, J.A., and Abdelzaher, T.: Range-free localization schemes for large scale sensor networks. In: *9th Annual International Conference on Mobile Computing and Networking*, 2003, San Diego, CA, pp. 81–95.

[HHS+99] Harter, H., Hopper, A., Steggles, P., Ward, A., and Webster, P.: The anatomy of a context-aware application. In: *5th International Conference Mobile Computing and Networking*, 1999, Seattle, WA, pp. 59–68.

[HS06] Huang, Q. and Selvakennedy, S.: A range-free localization algorithm for wireless sensor networks, *Proceedings of the 63rd IEEE Vehicular Technology Conference*, 2006, Melbourne, Australia.

[HWB00] Hightower, J., Want, R., and Borriello, G.: SpotON: An indoor 3D location sensing technology based on RF signal strength. In: *Technical Report, CSE 2000-02-02*, February 2000, University of Washington, Seattle.

[KH00] Koshima, H. and Hoshen, J.: Personal locator services emerge. *IEEE Spectrum* 37 (2000), 41–48.

[LLP06] Lanzisera, S., Lin, D.T., and Pister, K.S.J.: RF time of flight ranging for wireless sensor network localization, In: *4th Workshop on Intelligent Solutions in Embedded Systems*, 2006, Vienna, Austria.

[lor07] LORAN, http://www.navcen.uscg.gov/loran/, 2007.

[LR03] Langendoen, K. and Reijers, N.: Distributed localization in wireless sensor networks—A quantitative comparison. In: *Computer Networks: The International Journal of Computer and Telecommunication Networking,* 2003, Elsevier, New York, pp. 499–518.

[MH95] Murphy, W. and Hereman, W.: Determination of a position in three dimensions using trilateration and approximate distances. In: *Technical Report, MCS-95-07*, 1995, Colorado School of Mines, Golden, CO.

[NLLP03] Ni, L.M., Liu, Y., Lau, Y.C., and Patil, A.P.: LANDMARC: Indoor location sensing using active RFID. In: *IEEE International Conference on Pervasive Computing and Communications*, 2003, Dallas–Fort Worth, TX, pp. 407–415.

[NN01] Niculescu, D. and Nath, B.: Ad hoc positioning system (APS). In: *IEEE Global Telecommunications Conference*, 2001, San Antonio, TX, pp. 2926–2931.

[NN03] Niculescu, D. and Nath, B.: Ad hoc positioning system (APS) using AOA. In: *IEEE Annual Joint Conference Computer and Communications Societies*, 2003, San Francisco, CA, pp. 1734–1743.

[PBDT03] Priyantha, N.B., Balakrishnan, H., Demaine, E., and Teller, S.: Anchor-free distributed localization in sensor networks. In: *Technical Report, TR-892*, 2003, MIT Press, Boston, MA.

[PCB00] Priyantha, N., Chakraborty, A., and Balakrishnan, H.: The cricket location-support system. In: *6th Annual International Conference on Mobile Computing and Networking*, 2000, Boston, MA, pp. 32–43.

[PJ96] Parkinson, B. and Jr., J. S.: Overview of GPS operation and design. In: *Global Positioning System: Theory and Applications*, Bd. 1 and vol. 163 of *Progress in Astronomics and Aeronautics*, chapter 2, 1996, American Institute of Aeronautics and Astronautics, Washington, 1996, pp. 29–56.

[Rap02] Rappaport, T. S.: *Wireless Communications—Principles and Practice*, 2nd edn., 2002, Prentice Hall, Upper Saddle River, NJ.

[RBSJ79] Raab, F., Blood, E., Steiner, T., and Jones, H.: Magnetic position and orientation tracking system. *IEEE Transactions on Aerospace and Misc Systems* 15 (1979), 709–717.

[RBT06] Reichenbach, F., Blumenthal, J., and Timmermann, D.: Improved precision of coarse grained localization in wireless sensor networks. In: *9th Euromicro Conference on Digital System Design*, 2006, Dubrovnik, Croatia, pp. 630–637.

[RBTB06] Reichenbach, F., Born, A., Timmermann, D., and Bill, R.: A distributed linear least squares method for precise localization with low complexity in wireless sensor networks. In: *2nd IEEE International Conference on Distributed Computing in Sensor Systems*, June 2006, Springer Verlag, San Francisco, CA, pp. 514–528.

[Rei04] Reichenbach, F.: Positionsbestimmung in drahtlosen Ad-Hoc Sensor-Netzwerken, March 2004, Universität Rostock, Rostock Diplomarbeit.

[Rei07] Reichenbach, F.: Resource aware algorithms for exact localization in wireless sensor networks, University of Rostock, Rostock, Germany, PhD Thesis, 2007.

[Röm05] Römer, K.: Time synchronization and localization in sensor networks, ETH Zurich, Switzerland, PhD Thesis, 2005.

[SB05] Salomon, R. and Blumenthal, J.: Coarse-grained localization: Extended analyses and optimal beacon distribution. In: *10th IEEE Conference on Emerging Technologies and Factory Automation*, 2005, Catania, Italy, pp. 8–16.

[Sch04] Schiller, J.: Scatterweb—A platform for teaching and prototyping wireless sensor networks. 2004, Computer Systems and Telematics, Freie Universität Berlin, Germany.

[SG99] Stuber, G.L. and Gaffery, J.J.: Radio location techniques. In: *Mobile Communications Handbook*, 1999, CRC Press, Boca Raton, FL.

[Sha49] Shannon, C.E.: Communication in the presence of noise. *Proceedings of IEEE* 37 (1949), 10–21.

[SHS01] Savvides, A., Han, C.-C., and Srivastava, M.B.: Dynamic fine grained localization in ad-hoc networks of sensors. In: *7th ACM MobiCom*, 2001, Rome, Italy, pp. 166–179.

[SHS06] Stoleru, R., He, T., and Stankovic, J.A.: Range-free localization. In: *Chapter in Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks*, vol. 30, Springer, 2006.

[SPS03] Savvides, A., Park, H., and Srivastava, M.B.: The n-hop multilateration primitive for node localization problems. *Mobile Networks and Applications* 8 (2003), 443–451.

[SRB01] Savarese, C., Rabaey, J., and Beutel, J.: Location in distributed ad-hoc wireless sensor networks. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2001, Salt Lake City, UT, pp. 2037–2040.

[SRL02] Savarese, C., Rabaey, J., and Langendoen, K.: Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In: *USENIX Annual Technical Conference*, 2002, Monterey, CA, pp. 317–327.

[SS02] Simic, S. and Sastry, S.: Technical report, UC Berkeley, UCB/ERL M02/26, 2002.

[Sta96] International Organization for Standardization: Open systems interconnect—Basic reference model. In: ISO/IEC 7498-1:1994e, 2nd edn, 1996.

[Sto05] Stojmenovic, I.: Localization in sensor networks. *Handbook of Sensor Networks—Algorithms and Architectures*. In: Wiley Series on Parallel and Distributed Computing, 2005, John Wiley & Sons Inc., New York.

[Tec07] Nanotron Technologies: Real time location systems, 2007, Berlin, Germany.

[TRV+06] Turau, V., Renner, C., Venzke, M., Waschnik, S., Weyer, C., and Witt, M.: The Heathland Experiment: Results and Experiences, 2006, Chicago, IL.

[ubi07] Ubisense, http://www.ubisense.net/. 2007.

[Wan92] Want, R.: The active badge location system. *ACM Transactions on Information Systems* 10 (1992), 91–102.

[WL98] Werb, J. and Lanzl, C.: Designing a positioning system for finding things and people indoors. *IEEE Spectrum* 35 (1998), 71–78.

7

Power-Efficient Routing in Wireless Sensor Networks

7.1 General Remarks on Routing in Wireless Sensor Networks 7-2
Data Delivery Model • Direct or Multi-Hop Routing • Performance Metrics for WSN Routing Protocols • Role of Topology Management Protocols

7.2 Overview of Energy-Saving Routing Protocols for WSNs 7-4
Optimization-Based Routing Protocols • Data-Centric Routing Protocols • Cluster-Based Routing Protocols • Location-Based Routing Protocols • QoS-Enabled Routing Protocols

7.3 Data-Centric Power-Efficient Routing Protocols 7-6
Sensor Protocols for Information via Negotiation • Directed Diffusion Protocol • Rumor Routing Protocol • Constrained Anisotropic Diffusion Routing Protocol • Cougar Protocol • Two-Tier Data Dissemination Protocol

7.4 Optimization-Based Power-Aware Routing Protocols 7-13
Minimum-Cost Forwarding Protocol • Sequential Assigned Routing Protocol • Energy-Aware Routing Protocol • Maximum Lifetime Routing Protocol

7.5 Cluster-Based Energy-Efficient Routing Protocols 7-17
Low Energy-Adaptive Cluster Hierarchy Protocol • LEACH Extensions and LEACH-Inspired Protocols • Energy-Aware Routing Protocol in Cluster-Based Sensor Networks

7.6 Location-Based Energy-Aware Routing Protocols 7-28
Minimum Energy Communication Network (MECN) Protocol • Geographic and Energy-Aware Routing (GEAR) Protocol

7.7 Energy-Aware QoS-Enabled Routing Protocols..... 7-31
Energy-Aware QoS Routing Protocol for Wireless Sensor Networks • Real-Time Power-Aware Routing (RPAR) Protocol

7.8 Topology Control Protocols for Energy-Efficient Routing 7-33
Geographic-Adaptive Fidelity (GAF) Protocol • Span Protocol • Sparse Topology and Energy Management (STEM) Protocol

7.9 Summary and Open Issues 7-36

References 7-37

Lucia Lo Bello
University of Catania

Emanuele Toscano
University of Catania

7.1 General Remarks on Routing in Wireless Sensor Networks

A wireless sensor network (WSN) is a collection of nodes organized into a cooperative network, typically operating in an unattended environment. Each node is equipped with a processing element, a radio-frequency transceiver (usually with a single omnidirectional antenna), a number of sensors and actuators, memories (data, program, and flash), and a power source. Since the nodes are equipped with small, often irreplaceable, batteries with limited capacity, it is crucial that the network be energy-efficient in order to maximize its lifetime.

The nodes in a WSN are also capable of performing other functions such as data processing and routing.

Routing in WSNs is quite different from traditional routing in wired networks. Table 7.1 summarizes the main differences between WSNs and traditional distributed systems relying on wired connections, while Table 7.2 shows the factors which make the difference between routing in WSNs and traditional routing on the Internet.

Very simple approaches to relaying data in WSNs are Flooding and Gossiping [Hed88]. In flooding, each packet received by a sensor node is broadcast to all its neighbors, until the destination is reached. With a similar approach, if a path between the source and destination nodes exists, it will be found. However, as a large number of duplicate packets will be generated, a maximum hop count is needed in order to prevent infinite loops.

The Gossiping protocol works by selecting a random neighbor (other than the source of the incoming packet) at every hop, until the packet reaches its destination. Both Flooding and Gossiping are easy to implement, as they do not require topology information. However, these mechanisms have a few major drawbacks. One negative aspect is that they suffer from the well-known problem of data overlap between nodes belonging to the same area (as sensor nodes often cover overlapping areas, sensed data also overlaps). In addition, flooding algorithms also suffer from the implosion problem, which occurs when multiple nodes send the same data to the same destination node. The Gossiping protocol avoids the problem of implosion, but can cause very large end-to-end delays in data delivery, especially with large networks. Finally, the major drawback of both approaches is that they do not take power consumption into account.

TABLE 7.1 Features Which Make WSNs Very Different from Traditional Distributed Systems

Distributed Systems	WSNs
Wired	Wireless
Reliable connections	Error-prone connections (depending on the link quality, signal strength, noise, interference, atmospheric conditions, etc.)
Symmetric links	Asymmetric links
Unlimited power	Scarce power
Generally not real-time	Typically real-time
Each individual node is important	Aggregate behavior counts
Location-independent	Location-dependent
Resources not an issue	Resource-limited

TABLE 7.2 Factors Which Make the Difference between WSN Routing and Traditional Internet Routing

Variable Link Properties
Traffic patterns
Routing decisions usually based on geographic coordinates and/or data semantics, instead of node IDs
Unicast, area multicast or anycast semantics
Node sleep/wakeup for power management
Voids on the routing path

As sensor nodes are typically battery-operated, energy saving is a major design issue in WSNs. It has been proven that the communication cost for sensor nodes is much higher than the computational cost. For this reason, when deploying a WSN, the network topology, and thus the distance between communicating nodes, is a crucial aspect. In some cases sensors can be put in place in a controlled way, so the WSN can be built in an energy-efficient way if a suitable node placement strategy is followed. However, in most practical cases sensor nodes are randomly scattered over the field, so WSNs are self-organizing and deployed in an ad hoc fashion, and the network topology cannot be set according to any strategy targeting energy consumption. As a result, in order to prolong the network's lifetime as much as possible, approaches aiming at reducing energy consumption have to be implemented at all the different levels of the network protocol stack, from the physical up to the application layer, and even cross-layer approaches to save energy are found in the literature.

The strategies working at the physical layer try to reduce system-level power consumption through hardware design or by means of suitable techniques, such as dynamic voltage scaling or duty-cycle reduction. The approaches operating at the data link layer typically exploit low-power medium access control (MAC) protocols aimed at reducing the main causes of energy wastage, i.e., collisions, overhearing, idle listening, and the protocol overhead due to the exchange of a high number of control packets. At the network layer energy consumption is mainly dealt with in data routing. In Section 7.2, an overview of routing protocols for WSNs addressing energy saving and their classification into five categories are provided. However, before going into the classification of the different routing protocol categories, it is advisable to pinpoint some important aspects which influence the design and evaluation of routing protocols for WSNs, such as the data delivery model, the forwarding model, the performance metrics, and the role of topology management.

7.1.1 Data Delivery Model

According to the data delivery mode, WSNs may be classified as proactive or reactive. In proactive networks, which are those typically used for monitoring purposes, data delivery from sensor nodes to the sink is continuous, while in reactive networks it is the occurrence of an event or the reception of a query which triggers the transmission of data to the sink. The data delivery model has a significant impact on the performance of energy-efficient routing algorithms for WSNs, so approaches which are very advantageous for proactive networks do not work well with reactive ones, and vice versa.

7.1.2 Direct or Multi-Hop Routing

In WSNs routing can be direct or multi-hop. In direct routing, nodes transmit directly to the sink, while in multi-hop routing data is forwarded node by node toward the sink. As multi-hop routing imposes an overhead due to route management and MAC, when nodes are quite close to the sink direct routing is advisable. However, as the transmission power to be used to transmit data to a remote node increases with the distance between nodes, in WSNs consisting of a large number of sensors scattered over large areas multi-hop routing is the only viable option in order not to quickly drain the battery of the sender nodes. For the same reason, short-range hop-by-hop communication is to be preferred to long-range communication.

7.1.3 Performance Metrics for WSN Routing Protocols

In order to evaluate the effectiveness of WSN routing protocols, performance metrics such as the average delay per packet or delivery ratio may be used. However, as the most important objective is usually energy efficiency in the context of WSNs, metrics that measure the energy efficiency of routing protocols are mainly used. For instance, protocols that aim to minimize energy consumption may use the average consumed energy or the total energy dissipated as their principal metric to compare with

other protocols. An interesting approach is proposed in [Lin01], where the energy∗delay metric is suggested. As energy consumption strongly depends on the amount of data exchanged, in some cases the overhead due to route setup and management is also used to assess the performance of a WSN.

Another quite popular metric is the time to get the network partitioned. This is especially important in algorithms that aim to maintain network connectivity as long as possible. To assess their performance, optimization-based routing algorithms may use the same metric optimized by the cost function, e.g., the energy consumed along the chosen paths. However, the most important objective of a routing protocol is usually to maximize the network lifetime rather than minimize the energy of single paths. So the most widely used metrics refer to the network lifetime, which can be expressed in several different ways. For example, in WSNs used for intrusion or fire detection, where network quality decreases considerably as soon as one node dies, the first node dies (metric, which gives an estimated value for this event for a specific network configuration, is appropriate [HAN02]. In scenarios where the loss of a single or a few nodes does not automatically compromise network operation, on the other hand, the half nodes alive metric, which gives an estimated value for the half-life of a WSN, can be used. Finally, the last node dies metric gives an estimated value for the overall lifetime of a WSN. These metrics can be expressed in terms of either time or rounds, where such a parameter is defined (e.g., in Low Energy-Adaptive Cluster Hierarchy, LEACH [Hei00]).

Other metrics may depend on the protocol category, e.g., quality of service (QoS) metrics for QoS-enabled protocols, such as deadline hit and miss ratio for real-time routing.

7.1.4 Role of Topology Management Protocols

Energy consumption in WSNs is typically dominated by the node communication subsystems, and can only be significantly reduced by transitioning the embedded radios to a sleep state, at the expense of changes in the network topology. The role of topology management protocols is to select which nodes can turn off their radios without compromising the network capacity.

To this end they coordinate the sleep transitions of all the nodes, while ensuring adequate network connectivity, in such a way that data can be efficiently forwarded to the data sink.

The following parts of this chapter are organized as follows. Section 7.2 presents an overview and classification of energy-efficient routing protocols for WSNs. Sections 7.3 through 7.7 describe and discuss in detail notable representatives of power-efficient routing protocols. For ease of presentation, each protocol is presented in the context of the category it belongs to. In Section 7.8 topology management schemes for WSNs are addressed. Finally Section 7.9 outlines open issues in energy-efficient routing protocols for WSNs.

7.2 Overview of Energy-Saving Routing Protocols for WSNs

Energy-saving routing protocols for WSNs can be classified into five main categories, i.e., optimization-based, data-centric, cluster-based, location-based, and QoS-enabled. Such categories are not necessarily disjoint, and some examples of routing algorithms matching multiple categories can be found.

7.2.1 Optimization-Based Routing Protocols

A broad spectrum of routing algorithms for WSNs aiming at reducing the energy consumption of sensor nodes is present in the literature. Some of them take energy into account explicitly when routing sensor data, and for most of them the main goal is the optimization of some metric. For this reason, we will henceforward refer to them as optimization-based energy-aware routing (EAR)

protocols. Example of metrics to be minimized are the energy consumed per message, the variance in the power level of each node, the cost/packet ratio, or the maximum energy drain of any node.

Trying to minimize the energy consumed per message may lead to poor routing choices, as some nodes could be unnecessarily overloaded and thus could quickly extinguish their batteries. A more effective option, if all nodes are equally important for the WSN to operate correctly, is to try to balance the battery power remaining in the nodes, as there is no point in having battery power remaining in some nodes while the others have already run out of power.

Minimization of the cost/packet ratio involves labelling different links with different costs and then choosing the best option so as to delay the occurrence of network partitioning as long as possible. On the other hand, the idea of minimizing the maximum energy drain of any node derives from the consideration that network operations start to be compromised when the first node exhausts its battery, so it is advisable to minimize battery consumption in this node.

A number of optimization-based power-aware routing approaches try to maximize network lifetime. They target network survivability, meaning that their goal is to maintain network connectivity as long as possible. To achieve this goal, "optimal" routes that avoid nodes with low batteries and try to balance the traffic load are chosen [Cha00]. The use of optimization techniques to find the minimum-cost path, where the cost parameter takes energy (alone or combined with other metrics) into account, is proposed. However, the minimum cost path approach has a drawback in terms of network lifetime in the long term. In fact, a protocol which, once it has found an optimal path, uses only that path for routing, will eventually deplete the energy of the nodes along the path. As large differences in the energy levels of the WSN nodes could lead to undesired effects such as network partitioning, suitable solutions have been developed. A notable example is the EAR protocol [Sha02], where network survivability is pursued by choosing not a single optimal route, but a set of good routes, i.e., suboptimal paths which are selected in a probabilistic way. Section 7.3 will present and discuss this approach.

7.2.2 Data-Centric Routing Protocols

Unlike the optimization-based routing algorithms described above, other routing protocols for WSNs obtain low-power consumption for sensor nodes without explicitly dealing with energy considerations when performing route selection, but implementing mechanisms which reduce energy wastage. One of the main causes of energy wastage in WSNs is data redundancy, which derives from a combination of a lack of global identifiers (as no IP-like addressing is possible in WSNs) and the random deployment of sensors, which in many cases makes it difficult, if not unfeasible, to select a specified set of sensors within a given area. To solve this problem, data-centric routing approaches were introduced. In these approaches, data is named using high-level descriptors, called meta-data, and data negotiation between nodes is used to reduce redundancy. Another approach to reduce data redundancy (and the consequent energy wastage) is by performing data aggregation at the relaying nodes, which consists of combining data from different sources and eliminating duplicates, or applying functions such as average, minimum, and maximum. Data aggregation also overcomes the overlap problem, which arises when multiple sensors located in the same region send the same data to the same neighbor node. Thanks to data aggregation significant energy savings can be achieved, as computation at sensor nodes is less energy-consuming than communication. When performed through signal processing techniques, data aggregation is referred to as data fusion. According to the kind of routing protocol, data aggregation may be a task performed by special nodes or any node in the network. Notable examples of data-centric routing protocols which perform data aggregation for energy-saving purposes are sensor protocols for information via negotiation (SPIN) [Kul99] and Directed Diffusion [Int00], which in turn inspired several other protocols. They will be discussed in Section 7.3.

7.2.3 Cluster-Based Routing Protocols

Another critical aspect for energy consumption is the presence of nodes which, being either closer to the sink or on the optimal (e.g., minimum cost) path to the sink, perform more relaying than the other nodes, thus depleting their energy reserve faster than the others. When such nodes run out of energy, network survivability is compromised, and when all the nodes closest to the sink die, the sink itself becomes unreachable. To avoid this problem, hierarchical or cluster-based routing was introduced. In cluster-based routing, special nodes called cluster heads form a wireless backbone to the sink. Each of them collects data from the sensors belonging to its cluster and forwards it to the sink. In heterogeneous networks, cluster heads may be different from simple sensor nodes, being equipped with more powerful energy reserves. In homogeneous networks, on the other hand, in order to avoid a quick depletion of cluster heads, the cluster head role rotates, i.e., each node works as a cluster head for a limited period of time. Energy saving in these approaches can be obtained in many ways, including cluster formation, cluster-head election, etc. Some of these approaches also perform data aggregation at the cluster-head nodes to reduce data redundancy and thus save energy. Notable examples of cluster-based routing protocols are LEACH [Hei00] and its extensions, which will be discussed in Section 7.5.

7.2.4 Location-Based Routing Protocols

Location-based routing protocols use position information for data relaying. Location information can be exploited for energy-efficient data routing in WSNs as, based on both the location of sensors and on knowledge of the sensed area, a data query can be sent only to a particular region of the WSN rather than the whole network. This feature of location-based routing protocols may allow for a significant reduction in the number of transmissions and thus in the power consumption of sensor nodes. Location-based routing protocols will be discussed in Section 7.6.

7.2.5 QoS-Enabled Routing Protocols

A number of routing algorithms for WSNs which take some kind of QoS into account have been proposed. QoS is usually addressed in terms of either end-to-end or average delay, or deadline miss ratio. Some of these protocols are energy-aware, while others are not. A notable non-energy-aware QoS-enabled protocol is SPEED [He03] [He05], a well-known protocol which combines feedback control and nondeterministic geographic forwarding to achieve a predictable end-to-end communication delay under given distance constraints. The basic idea of SPEED is to maintain a desired delivery speed across the sensor network. Although SPEED is not energy-aware, it inspired a number of energy-aware QoS-enabled routing protocols for WSNs, some of which will be described in Section 7.7.

7.3 Data-Centric Power-Efficient Routing Protocols

7.3.1 Sensor Protocols for Information via Negotiation

The aim of the family of adaptive negotiation-based protocols for WSNs called SPIN, presented in [Kul99], is to efficiently disseminate information among sensors in an energy-constrained environment. The basic idea in SPIN is to name the data using high-level descriptors called metadata. The use of metadata negotiation reduces the transmission of redundant data throughout the network, as compared to the classic Flooding protocol. Hence, SPIN protocols address major problems of flooding, i.e., message implosion, overlap, and resource blindness (as SPIN protocols are energy-aware). This is done by negotiating data at the sensor nodes before transmission occurs and introducing

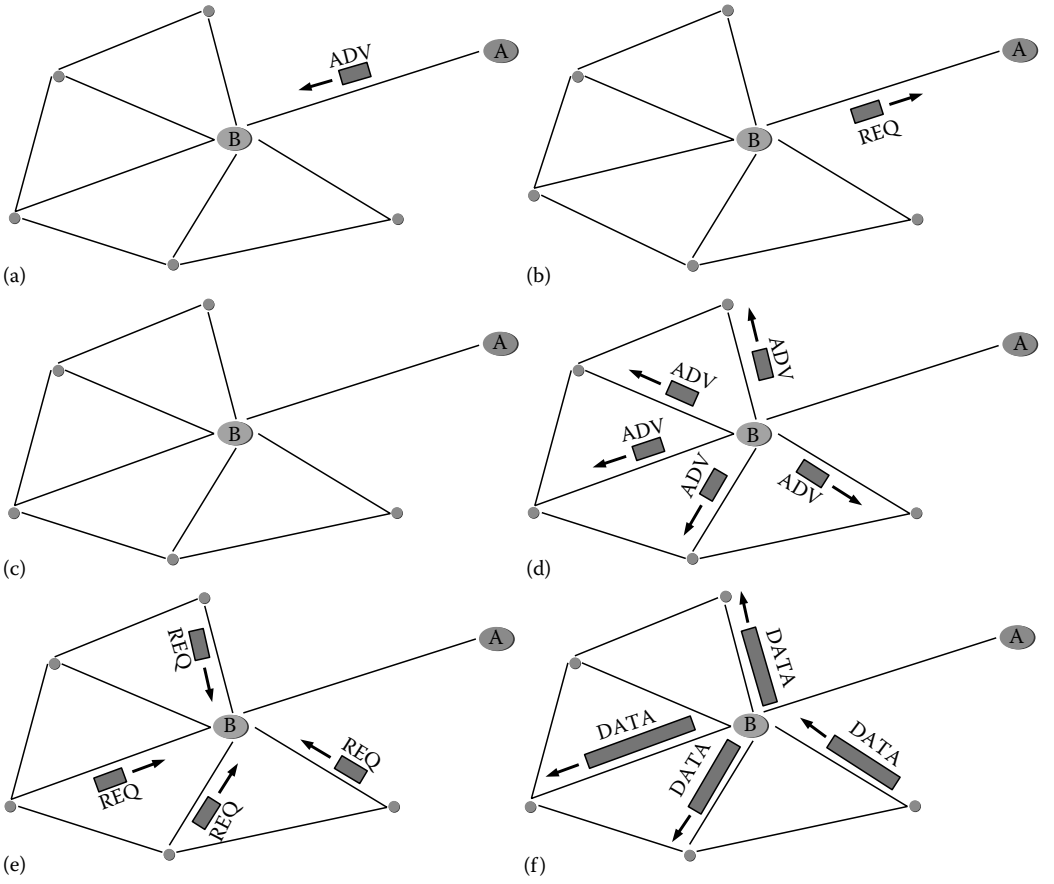


FIGURE 7.1 SPIN protocol. (a) Node A advertises its data to node B. (b) Node B in response sends a request to node A. (c) Node A sends the requested data to node B. (d) Node B sends an advertisement to (ADV) to its neighbors. (e) The interested nodes respond with request messages. (f) Data is transmitted by node B. (Redrawn from Kulik, J., Rabiner, W., and Balakrishnan, H., Adaptive protocols for information dissemination in wireless sensor networks, in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 174–185, 1999.)

resource adaptation. In SPIN each sensor node features a resource manager that keeps track of power consumption and triggers energy-aware behavior.

SPIN does not specify a standard metadata format, as it is application-specific. However, as metadata exchange is done before data transmission, in order to maintain energy efficiency the size of metadata should be significantly less than that of sensor data.

Three types of messages are introduced by SPIN protocols.

- ADV messages, which allow a sensor to advertise that a node has data to share. This message contains the metadata used to identify the sensor data.
- REQ messages, which allow a node to request specific data.
- DATA messages, which contain the actual sensor data.

The basic functioning of the SPIN protocols is represented in Figure 7.1.

As is shown in Figure 7.1, SPIN works in three stages. When a node has new sensor data, it sends an ADV message to its neighbors with the relative metadata (ADV stage). When neighbors receive the ADV message, they check whether the advertised data has already been received (or requested). If not, the node sends a REQ message to the source (REQ stage). Finally, the source of the ADV message responds to the REQs with a DATA message, containing the advertised data. Upon receiving data, a node can apply data aggregation techniques and then advertise the aggregated data. Furthermore, a low-energy threshold can be used to reduce participation in the protocol, i.e., a node will participate in a stage only if its remaining energy is sufficient to complete the following stage.

As compared to classical Flooding and Gossiping protocols, SPIN features shorter dissemination times and higher reliability. As the power consumption of the nodes during the transmission of DATA packets is usually much higher than that needed for ADV/REQ packets, SPIN also provides lower energy consumption than flooding. However, the advantage in terms of energy consumption becomes less pronounced when the amount of data is small. In fact, the energy reduction technique adopted here decreases the amount of data to be transmitted. This is effective when data transmission requires more power than reception and idle states. However, as discussed in [Abi00], this assumption is not always true for sensor nodes. For this reason, a number of cross-layer routing approaches exist, such as LEACH [Hei00], which directly manage the low-power/sleep states of the nodes, thus decreasing their duty cycles.

7.3.2 Directed Diffusion Protocol

Directed Diffusion, presented in [Int00], represents one of the most important data dissemination paradigms for WSNs, as it introduces a naming scheme, in which data generated by sensor nodes is named by attribute–value pairs. In order to save energy, short-range hop-by-hop communication is preferred over long-range communication to the destination, and data aggregation is performed locally to reduce the data size before transmission.

The basic idea of Directed Diffusion is that nodes request data by sending interests. An interest specifies a list of attribute–value pairs that describe the sensing task, i.e., type, interval, duration, sensing area, etc. Data matching the interest is drawn toward the node itself.

Both sensing tasks and data sent in response to interests are named using a similar naming scheme, based on attribute–value pair description, i.e., name and type of objects, data rate of events, duration, geographical area, etc.

For each sensing task, the sink node (i.e., the node that originated the query) periodically broadcasts an interest message to its neighbors. Each node has an interest cache, where several parameters such as a time stamp, a gradient, the data rate, duration, etc. are maintained for each distinct interest.

Each gradient is a reply link to the neighbor from which the interest was received and contains a data rate field (derived from the interval attribute of the interest) and a duration field (derived from the time stamp and *expiresAt* attributes of the interest) indicating the approximate lifetime of the interest.

Such parameters are updated every time a node receives an interest. Interest entries do not contain information about the sink node, as data is delivered through the gradients, which specify the addresses of the neighbors from which the interest has been received. As the nodes maintain only local information and no topology information is needed, the scalability of Directed Diffusion is high. In order to spread interests throughout the network, after an interest has been received a node may decide to resend it to all or to some of its neighbors. Each node also has a data cache used to direct interests (thus avoiding flooding) and to prevent transmission loops.

When a node receives a data message from a neighbor node, it first attempts to find a matching interest in the interest cache. If a matching interest is found, the message can be forwarded to each node for which it has a gradient, otherwise it is silently dropped.

With such a mechanism, the sink may receive the same data several times from different paths. For this reason a reinforcement scheme has been introduced, in order to use network bandwidth more efficiently and thus to save energy. The reinforcement mechanism makes use of the data rate parameter of the interest. In fact, at the beginning, the sink node broadcasts a low data rate interest. When the gradient has been set up, i.e., when the sink starts receiving low data rate messages, it reinforces one particular neighbor by increasing its data rate. When a node realizes that it has been reinforced, it also has to reinforce at least one neighbor. From the sequence of reinforcements on single hops, a high data rate path is established from the source of the events to the sink node.

The functioning of the Directed Diffusion routing protocol is shown in Figure 7.2.

A convenient rule for the selection of the node to be reinforced has to be adopted in order to select a low delay path. For example, a node might choose the neighbor from which more events have been received, or the neighbor that regularly reports events before others. However, the path selected is generally not optimum.

There is also a mechanism for negative reinforcements that change a high data rate to a low rate, which can be used when a better path is found.

The reinforcement mechanism can also be used locally to repair failing or degraded paths. In this case, the reinforcement is not triggered by the sink node, as an intermediate node can reinforce an alternative link. However, this mechanism has to maintain alternative low data rate paths, so there is a trade-off between robustness and energy efficiency.

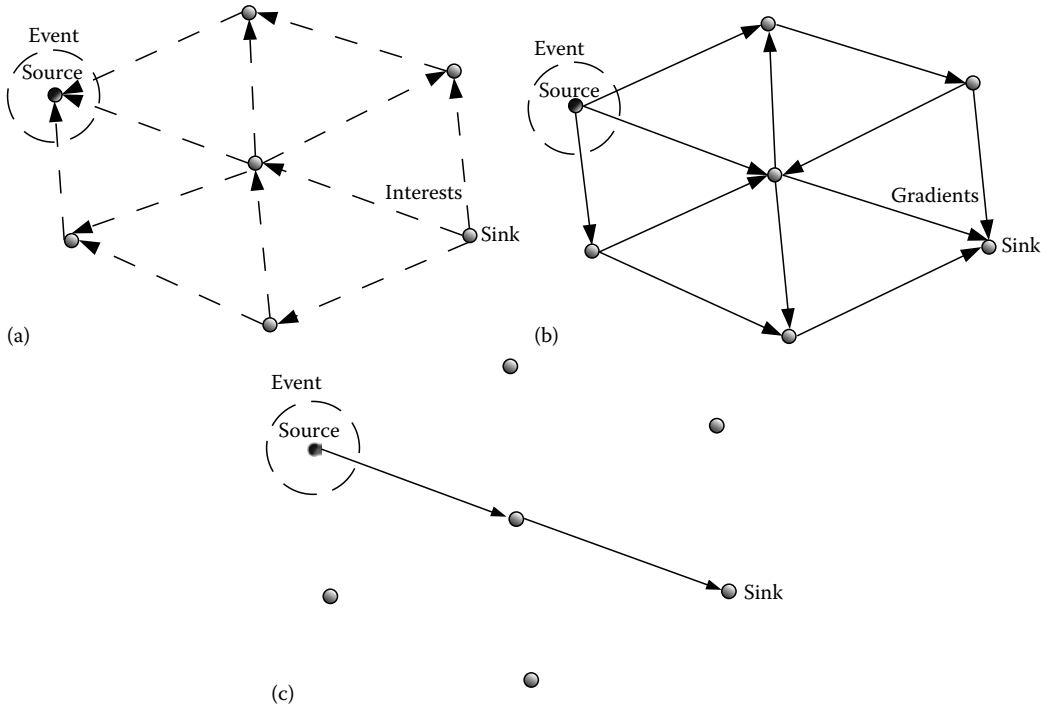


FIGURE 7.2 Directed Diffusion protocol operations. (a) Interest propagation. (b) Initial gradients setup. (c) Data delivery along reinforced path. (Redrawn from Intanagonwiwat, C., Govindan, R., and Estrin, D., Directed diffusion: A scalable and robust communication paradigm for sensor networks, in *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 56–67, 2000.)

Directed Diffusion does not directly decrease the duty cycle of the nodes (i.e., nodes do not go to sleep), so energy efficiency is only achieved by reducing the number of packets transmitted and performing data aggregation. This technique is efficient when the idle power consumption is much lower than that required for transmitting and receiving. However, in many real cases of devices characterized by high power consumption in idle states (i.e., with CSMA protocols) Directed Diffusion does not achieve high energy savings. Despite this, with respect to the SPIN protocol family, Directed Diffusion features lower overheads in terms of packets sent, as SPIN needs explicit negotiation for each packet, while Directed Diffusion uses periodic interests and gradients perform implicit negotiation. Lower overheads also imply higher energy efficiency. However, in Directed Diffusion the number of data packets is higher than in SPIN, due to the existence of multiple paths, so the energy consumption of Directed Diffusion also depends on the size of data packets as compared to negotiation packets.

In [Gad06] the Directed Diffusion approach was compared with classical MANET routing protocols, such as the Ad-hoc On-demand Distance Vector protocol (AODV) [Perk99] and the Optimized Link State Routing protocol (OLSR) [Jacq01], and with another data-centric protocol, Two-Tier Data Dissemination (TTDD) [Ye02], for both performance and energy efficiency. Simulation results show that AODV regularly outperforms others in terms of packet delivery ratio, latency, and energy efficiency. The lower efficiency of the Directed Diffusion protocol is probably due to the higher data redundancy. In fact, although there is only one reinforced path, multiple low data rate links are maintained. In addition, the scenarios tested only comprise a small number of nodes producing data, so it is more suitable for classical ad-hoc networks than for WSNs. On the other hand, WSNs may comprise a very large number of sensing nodes. In such scenarios, maintaining a routing table with an entry for each data flow may be too expensive for resource-constrained nodes such as typical WNS nodes, so the scalability of localized* WSN protocols (such as data-centric or location-based approaches) is required [Estr99]. Furthermore, WSN nodes are generally more faulty than other networks, as the network may operate in harsh environmental conditions. In such conditions, the Directed Diffusion approach can provide higher fault tolerance as multiple paths are maintained for each data flow.

The Directed Diffusion approach can definitely be useful for query-based sensor networks, as semantic information is used for forwarding. The drawback of this approach is that the naming scheme is strongly application-specific. However, a WSN is not usually a general-purpose network, so this is not a main concern.

7.3.3 Rumor Routing Protocol

Rumor Routing [Bra02] is a probabilistic data-centric routing protocol for large-scale WSNs containing several thousands of nodes, which is inspired by the Directed Diffusion approach. The Directed Diffusion approach disseminates queries through networks by means of (controlled) flooding. This is efficient if the amount of data to be transmitted is not small or if there are a large number of events. In some cases, i.e., when there are few events and many queries, event broadcast may be more efficient than query broadcast. Rumor Routing is something in between event broadcast and query broadcast, so it can be useful in the middle of the region depicted in Figure 7.3.

In Rumor Routing, each node maintains a neighbor table as well as an event table. When a node notices an event, it probabilistically generates an agent. An agent is a long-lived packet that travels through the network in order to propagate information about the sensed events to distant nodes. When a node receives such a packet, it updates the event table with the distance and the forwarding

* Localized protocols maintain and use only information about neighbor nodes; no overall network or transmission flow knowledge is required.

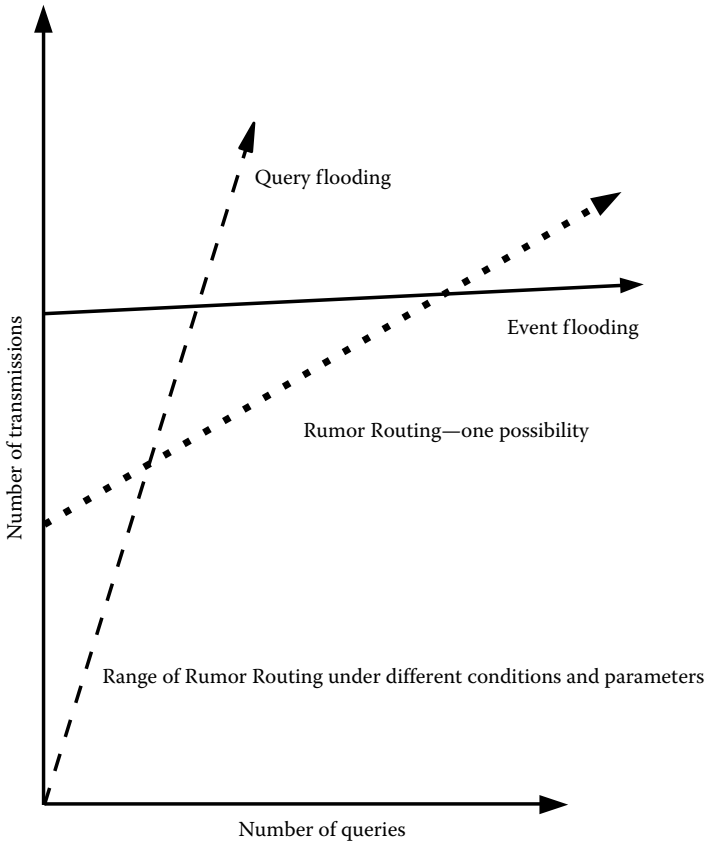


FIGURE 7.3 Rumor Routing compared to query flooding and event flooding. (Redrawn from Braginsky, D. and Estrin, D., Rumor algorithm for sensor networks, in *Proceedings of the First Workshop on Sensor Networks and Applications (WSNA)*, Atlanta, GA, October 2002.)

choice for the event. The generation probability is a parameter of the protocol, as well as the maximum hop number for an agent. When a node generates a query, if it already has a route to the source of the event it will use that route; otherwise it will forward the query in a random direction, until the query reaches a node that has observed the event (and therefore has a route) or the packet reaches the maximum number of hops allowed. With a similar approach, a query could fail to find an event. In this case, the query could be retransmitted or can be flooded through the network. When an event is found via random forwarding, the cost of flooding is avoided; otherwise this cost has to be added to the cost of random forwarding. Because of random forwarding, latency in this protocol may be high. Moreover, if random forwarding fails, the additional delay for flooding may be needed. So this protocol is definitely not suitable for time-critical applications.

Similarly to Directed Diffusion, Rumor Routing achieves energy efficiency by reducing the number of messages exchanged and performing data aggregation along the path. However, as Directed Diffusion performs query flooding, when the number of queries is high Rumor Routing performs better. Simulation results show that with an accurate selection of the protocol parameters Rumor Routing can outperform both event flooding and query flooding, but the downside is that performance is highly variable with varying parameters such as the number of agents and time-to-live values. As a result, in order to achieve high performance the design space has to be analyzed through extensive simulations, run using several possible configurations.

7.3.4 Constrained Anisotropic Diffusion Routing Protocol

In [Chu02] two techniques for querying and routing in WSNs are introduced, i.e., information-driven sensor querying (IDSQ) and CADR. The problem addressed here is how to perform queries and route data maximizing the information gain while minimizing latency and bandwidth utilization. The main idea of IDSQ/CADR is to introduce an information utility measure based on the estimation theory that models the information content and the spatial configuration of a network. In this way, a node can spread a query based on the evaluation of an objective function that considers both information and cost, and it can forward data based on the local gradient of the objective function. IDSQ/CADR can be viewed as a generalization of the diffusion approach, in which both information gain and communication cost direct data diffusion. In particular, IDSQ is a sensor selection algorithm that aims to find the optimal order of sensor queries that provide maximum information gain while balancing energy cost. The CADR protocol, on the other hand, determines the optimal routing path from the querying to the queried sensor (through the gradient of the objective function). This approach allows sensors to send packets only when there is interesting data to report. Moreover, only the part of the network with a better information/cost trade-off may be active. So this approach is more energy-efficient than Directed Diffusion, where queries are diffused in an isotropic fashion with (controlled) flooding over the entire network. Also, compared with approaches that only minimize the energy consumption for a single path, CADR achieves better performance, as both energy cost and information gain are taken into account through an appropriate utility function.

7.3.5 Cougar Protocol

In [Yao02] a data-centric protocol that models WSNs as a distributed database is presented. The Cougar protocol aims to define sensor tasks through declarative queries.

In order to achieve declarative queries in sensor networks, nodes have to implement a query layer between the network and application layers, which basically consists of a query proxy that performs in-network processing. In fact, unlike typical WSN applications, where data is forwarded and then analyzed, the Cougar approach moves part of the data analysis inside the WSN. In this way it is possible to reduce the amount of data to be transmitted. Since for WSN nodes local computations generally require less energy than data transmissions, this approach achieves better energy efficiency than centralized data extraction.

A query optimizer is located in the sink node, which produces an efficient query plan that reduces resource usage and so extends network lifetime. In order to generate a good plan, network conditions have to be known. For this purpose, a catalog can be created at the server, which maintains the useful information (that needs to be updated as the network parameters change). In addition to the network condition, the optimizer also considers existing query workload and tries to merge similar queries. A special node, the leader, is elected. This is the node where the computations of the aggregate values will take place. Then two query plans are generated, one for the leader and one for the other nodes, i.e., nodes send the leader local data from sensor or partially aggregated data, and the leader waits until the required data is received and then sends the overall aggregated values. This approach can be used for long-term queries, for example the average temperature value can be monitored for a long time and only updates for significant changes could be transmitted. Thanks to such a data aggregation, the Cougar approach is efficient for bandwidth and energy management, although it also has some drawbacks. For example, as a sensor has to wait to receive results to be aggregated in order to perform efficient data aggregation, it requires synchronization between sensor nodes along the communication path. This could be a problem because with high loss rates broken links may be hard to distinguish from long delays. In addition, this protocol requires a catalog that has to be updated at every change in the network (i.e., sensor position, connectivity, workload, etc.), and in a network with several thousands of nodes this is not a simple task.

7.3.6 Two-Tier Data Dissemination Protocol

Ye et al. addresses in [Ye02] the problem of data dissemination in the presence of mobile sink nodes. This protocol is designed for large-scale reactive sensor networks, where stationary and location-aware sensor nodes may detect target events, and multiple mobile sink nodes collect information while moving across the network. The TTDD protocol achieves scalable and efficient dissemination for query and data by proactively creating a grid structure when an event is detected. As sensor nodes are location-aware, a node with coordinates (x, y) that has sensed an event can set itself as a crossing point in the grid and send announcements to elect as dissemination nodes those closest to positions $(x \pm \alpha, y \pm \alpha)$, where α is the side of each cell. This can be achieved by using a greedy geographic forwarding scheme, i.e., a routing algorithm that chooses the neighbor closest to the destination. These announcements are sent recursively until the grid has been fully created. Once the grid has been set up, sinks can flood queries to their cell, i.e., the first tier. However, flooding is confined within the cell. Once the query reaches the closest dissemination node, it will be propagated toward the source node through the second tier, that is, the set of dissemination nodes. This mechanism significantly reduces the number of messages and hence energy as compared to standard query flooding. During this process, dissemination nodes store the location of nodes from which they receive queries. This information is used for subsequent data forwarding from the source to the sink node. In fact, data is forwarded through the reverse path until the cell of the sink is reached. Once it reaches that cell, a trajectory forwarding scheme is used to find the moving sink. According to this scheme, the sink includes in its queries the location of a designated sensor node, called a primary agent. This node represents the mobile sink for the immediate dissemination nodes. As the sink moves within the cell, the location of a close sensor node, the immediate agent, is updated. The immediate agent is initially the primary agent. So data is forwarded from the immediate dissemination node to the primary agent. If the sink has moved, this node will relay data to the new immediate agent. This node directly sends data to the sink. When a sink moves out of the cell, it may pick a new primary agent and re-flood the query.

The energy consumption problem is addressed here by dynamically creating a virtual grid in which only elected nodes perform data forwarding. In addition, query flooding is performed only within the grid, so both bandwidth and energy are saved. Compared with Data Diffusion [Int00], TTDD features reduced energy consumption when the number of sinks is low. However, its power consumption rapidly increases when the number of sinks increases, so when the number of sinks is large, Directed Diffusion could perform better in terms of energy consumption. However, TTDD also features lower delays when there are many sources and provides support for sink mobility.

7.4 Optimization-Based Power-Aware Routing Protocols

7.4.1 Minimum-Cost Forwarding Protocol

In [Ye01] Ye et al. propose an algorithm aiming to provide message delivery through the minimum-cost path from a sensor node to the sink in a large-scale WSN. This work tries to explore a new scalable solution to the minimum-cost forwarding problem, i.e., the cost field-based approach. The cost field for packets is something similar to the gravity field for water: as water flows from high to low posts, once the cost field is established, packets flow from source to sink nodes through the minimum-cost path. In order to achieve this behavior, each message has to keep the minimum required cost from the source to the sink node as well as the consumed cost from the source to the current node. The sender always uses broadcast packets. Intermediate nodes that receive the packet forward the message only if the consumed cost of the packet plus its own minimum cost to the sink is equal to the minimum cost specified by the source node. Hence, only nodes belonging to the minimum-cost path from the source to the sink perform data forwarding. Each node has to maintain only the minimum cost from

itself to the sink. As this approach does not need to maintain explicit path information for each packet or flow, it scales well with the number of data flows. However, before using this algorithm the cost field needs to be established.

Establishment of the cost field is started by the sink node, which broadcasts an ADV packet with a 0 cost. Each node N that receives an ADV from a neighbor M computes the cost from itself to the sink using this message, i.e., $L_M + C_{N,M}$, where L_M is the cost for the node M and $C_{N,M}$ is the cost from M to N. If it is lower than the current L_N value, the L_N value is updated and a back-off timer is scheduled (or reset). When the timer expires, an ADV message containing the L_N value is broadcast.

This protocol is efficient as long as the network is static. However, if there is a change in the topology, i.e., a node runs out of energy, the cost field has to be reestablished. Fault tolerance is low, because when a node fails many sources will not be able to forward data to the sink until the cost field has been reestablished. Although this protocol could be used with a generic cost function, the authors propose the use of energy as the path cost. So this protocol guarantees that the selected hop is always the optimum in terms of depleted energy. But while this approach achieves optimal results in terms of energy consumption, it does not optimize the network lifetime, as packets from the same node always use the same path. Load and energy consumption are thus not balanced so the most used node can quickly drain its battery.

7.4.2 Sequential Assigned Routing Protocol

The sequential assigned routing (SAR) protocol [Soh00] is a table-driven, multi-path routing protocol. As compared to classical ad-hoc protocols for MANET such as the AODV or the temporally-ordered routing algorithm (TORA)*, SAR tries to improve energy efficiency and fault tolerance in low-mobility networks such as WSNs while maintaining a desired QoS. The main idea of the SAR protocol is to route packets along multiple paths depending on the energy and QoS requirements. In addition to energy and required QoS, the SAR protocol also takes packet priority into consideration.

In order to set up multiple paths from each node to the sink, multiple trees are built. Each tree is rooted from a one-hop neighbor of the sink. Once the trees are built, most of the nodes will belong to more than one tree, so they will have multiple disjoint paths from which the sink node can be reached.

Each path is assigned two parameters by each node: a QoS metric and an energy resource, that is, the number of packets to be sent through that path before the node energy runs out. Path selection is performed by the source node, through computation of a weighted QoS metric, obtained from the additive QoS metric multiplied by a weight coefficient depending on the packet priority. Notice that by choosing the path, the one-hop neighbor of the sink is also selected. Furthermore, having different paths to the sink, in the event of failure a node can perform failure recovery, thus also improving fault tolerance. However, as the available energy of the nodes changes with time, it has to be periodically updated. In general this protocol is efficient, but it needs a high overhead to build the trees, maintain the tables, update the state, etc. The simulation results in [Soh99] show that the SAR protocol features better performance than algorithms that always select the minimum-energy path regardless of network conditions. The approach used to achieve energy efficiency is to optimize a weighted metric which takes the residual energy of nodes into account. This should achieve balanced energy consumption and an improved network lifetime. To further increase energy saving, the authors

* AODV is a source-initiated routing protocol that performs on-demand route discovery and maintenance, and relies on sequence numbers assigned by the destination to avoid loops. TORA is a link reversal algorithm that builds a directed acyclic graph (DAG) rooted at the destination in which data flows from higher to lower nodes.

recognize the need to lower the duty cycle of the nodes by dynamically powering the radio on and off. For this reason, in the same paper they also propose the use of self-organizing medium access control for sensor networks (SMACS), which is a MAC protocol that allows nodes to discover neighbors and create a transmission schedule to communicate in a time division multiple access (TDMA) fashion. Each link is assigned a time slot and a random frequency band. In this way SMACS can decrease the duty cycle of the nodes as well as the likelihood of collisions.

7.4.3 Energy-Aware Routing Protocol

The idea behind the energy-aware routing (EAR) protocol [Sha02] is that always using the lowest-energy path to route data may not be the optimal choice to optimize the WSN lifetime in the long term. In fact, a routing protocol that finds an optimal path and uses only that will quickly drain the energy of the nodes on that path. This would lead to a large disparity in the energy levels of the nodes, with the risk of network partitioning. In order to enhance network survivability, that is, to maintain network connectivity as long as possible, the EAR protocol does not find a single optimal route but a set of good routes, and probabilistically chooses one of them. In this way, a different path is used at different times, so there is a balance in the energy of different nodes and the whole network lifetime increases. This is similar to Directed Diffusion, as data flows once again along different paths. However, while Directed Diffusion constantly sends data along each path with different data rates, here only one path at a time is used.

The protocol comprises three different phases: setup, data communication, and route maintenance. During the setup phase the routing tables are created, with all the routes from sources to destinations and the relative energy costs. Connections are initiated by the destination nodes by localized flooding, i.e., request packets are forwarded to all neighbors that are closer to the source node than the node itself. This mechanism requires a location subsystem on each node. When a request is received (at each hop), the total cost of the path is updated by adding the energy metric for the neighbor that sent the request. That is, if the request is sent from node N_i to node N_j, N_j calculates the cost of the path as

$$C_{N_j,N_i} = \text{Cost}(N_i) + \text{Metric}(N_j, N_i). \tag{7.1}$$

The energy metric proposed in [Sha02] from node N_i to node N_j is $C_{ij} = e_{ij}^{\alpha} R_i^{\beta}$, where e_{ij} is the energy used for transmitting and receiving on the link, R_i is the normalized value for the residual energy, and α and β are two weighting factors.

Paths having a very high cost are discarded, while low-cost neighbors are added to the forwarding table on N_j. In the forwarding table, FT_j, each neighbor is assigned a probability that is inversely proportional to the cost, that is:

$$P_{N_j,N_i} = \frac{1/C_{N_j,N_i}}{\sum_{k \in FT_j} 1/C_{N_j,N_k}}. \tag{7.2}$$

Then N_j can calculate the average cost to reach the destination by probabilistically choosing the neighbors in the forwarding table FT_j, that is:

$$\text{Cost}(N_j) = \sum_{i \in FT_j} P_{N_j,N_i} C_{N_j,N_i}. \tag{7.3}$$

The average cost just calculated is set in the cost field of the request and forwarded toward the source node.

In the Data Communication phase of the protocol, each node simply forwards data packets to a random node in its forwarding table according to the stored probabilities, until the data packet

reaches the destination. Route maintenance is very simple, as it just sporadically performs localized flooding in order to keep all the paths alive.

EAR requires a lower overhead than Directed Diffusion. Simulation results show that the improvement in power consumption is up to 21.5%, while the network lifetime, here meaning the time when the first node runs out of energy, increases up to 44%. However, these results assume that energy is only consumed for packet transmission and reception, i.e., there is no energy consumption during the idle state. Thus, also in EAR, energy efficiency is accomplished by reducing the overhead. While the routing protocol does not explicitly decrease either the duty cycle or collisions, the authors propose the design of a sensor node that follows the PicoRadio architecture [DaS01], which features two radio transceivers, one always active but with a very low bit rate and power consumption, and the other with a low duty cycle and a higher data rate. With a similar node, CSMA/CA can be used in the low-power signalling channel in order to avoid collisions during data transmission.

7.4.4 Maximum Lifetime Routing Protocol

In [Cha00] the routing problem in a WSN is modeled as a linear optimization problem. Instead of trying to minimize the energy of single transmissions or single paths, the objective is maximization of the lifetime of the overall sensor network. This work extends previous work by the same authors presented in [Cha09] and [CT00]. In these works it was found out that, in order to maximize network lifetime, it is more useful to balance the energy consumption between the nodes in proportion to their energy rather than minimize the power consumed by transmissions. The algorithm used to balance energy along the whole path is the Maximum Residual Energy Path. This algorithm always selects the path with the maximum residual energy to route packets across the network. Nodes can calculate the maximum residual energy path in several different ways. In [CT00] the use of a path length vector is proposed, which works as follows: for each path from a source i and a destination d, a path length L_p is defined that is a vector of link costs c_{jk}, where (j,k) is a link in the path. The value of c_{jk} is the reciprocal of the residual energy at node j after the routing has been performed, i.e.,

$$C_{jk} = \frac{1}{E_j - e_{jk}} \tag{7.4}$$

where

E_j is the residual energy at node j

e_{jk} is the energy consumed to transmit over the link (j, k)

Using this formulation, the maximum residual energy path can be calculated as the shortest path with a slightly modified version of the Bellman–Ford algorithm [Bel58] [For62] which compares the entries of the path length vector in lexicographical order. In [Cha00] an alternative way to calculate the maximum residual energy path is also provided, simply using the largest element of the path length vector for comparison. As an alternative, the link cost is proposed, which reflects the number of packets that can be delivered with the residual energy of the nodes, so the cost value is

$$C_{ij} = \frac{e_{ij}}{E_i}. \tag{7.5}$$

The protocol has been compared with the minimum transmitted energy (MTE) algorithm [She98] (in which each node sends a message to the closest node on the way to the base station, BS) that uses e_{ij} as the transmission cost. Simulation results show that the Maximum Residual Energy algorithm is more energy-efficient than MTE, as the network lifetime noticeably increases. Furthermore, the new metric is shown to perform better than the one presented in [CT00], but the use of the maximum cost instead of the whole vector of path costs causes a slight performance degradation. So the best combination is the new metric combined with the old way to calculate the maximum residual energy path. However,

both the algorithm and the simulations only consider energy consumption due to transmission, i.e., no idle energy consumption or the possible use of low-power/sleep states are taken into account. While this protocol achieves interesting results from the theoretical point of view, in a real network there are also some drawbacks. For example, it lacks flexibility, as packets could be of variable size and the MAC layer could have some nondeterministic behavior, so it is not always simple to accurately estimate the energy required by each packet. In addition, this protocol requires topology knowledge, which in large WSNs is not simple to achieve.

7.5 Cluster-Based Energy-Efficient Routing Protocols

7.5.1 Low Energy-Adaptive Cluster Hierarchy Protocol

The LEACH protocol [Hei00] is a milestone for cluster-based protocols for WSNs, which has the objective of reducing power consumption by decreasing the transmission power and duty cycle of sensor nodes. The protocol is designed for WSNs where the BS is fixed and distant from the sensor nodes. In addition, the nodes are assumed to be homogeneous and energy-constrained. The design principles of the LEACH protocol are

- Localized coordination and control for both setup and operational phases
- Random rotation of cluster heads
- Data aggregation techniques to reduce communication costs

Nodes are grouped in clusters, each containing a cluster-head node. Non-cluster-head nodes only transmit to their cluster heads, while cluster heads perform long-distance transmission directly to the sink node.

The authors assume a first-order radio model. In this model, a radio dissipates $E_{\text{elec}} = 50\,\text{nJ/bit}$ to run the transmitter or receiver circuitry and $\varepsilon_{\text{amp}} = 100\,\text{pJ/bit/m}^2$ for the transmitter amplifier. The radios have power control and can consume the minimum required energy to reach the intended destinations. The radios can be turned off to avoid receiving unintended transmissions. An r^2 energy loss is used due to channel transmission.

The equations to calculate transmission costs and receiving costs for a k-bit message and a distance d are as follows:

Transmission

$$\begin{aligned} E_{\text{Tx}}(k,d) &= E_{\text{Tx-elec}}(k) + E_{\text{Tx-amp}}(k,d) \\ E_{\text{Tx}}(k,d) &= E_{\text{elec}} * k + \varepsilon_{\text{amp}} * k * d^2 \end{aligned} \tag{7.6}$$

Receiving

$$\begin{aligned} E_{\text{Rx}}(k) &= E_{\text{Rx-elec}}(k) \\ E_{\text{Rx}}(k) &= E_{\text{elec}} * k \end{aligned} \tag{7.7}$$

As receiving a message is not a low-cost operation, the number of receptions and transmissions for each message should be minimized.

According to this model, direct transmissions to the BS can be more efficient than the minimum transmission energy (MTE) routing protocol (in which each node sends a message to the closest node on the way to the BS) if the ratio $E_{\text{elec}}/\varepsilon_{\text{amp}}$ and the number of hops is sufficiently high. However, the cluster-based approach is more efficient than direct transmissions, as only a small percentage of nodes, i.e., the cluster heads, need to perform highly expensive long-distance transmissions to the sink. The majority of nodes can instead perform less expensive transmissions, as they only need to communicate with the cluster head, which is usually much closer than the sink. On the other

hand, cluster heads always make long-distance transmissions. Therefore if these nodes were fixed for the whole network lifetime as in classical clustering algorithms, they would drain their batteries quickly, preventing the nodes in all clusters from transmitting their data. For this reason, LEACH introduces random rotation of the cluster heads, so in the long term the energy distribution over the whole network will be fair. In addition, LEACH embeds other mechanisms to reduce energy consumption, such as data fusion inside each cluster and duty-cycle reduction for non-cluster-head nodes, which shut down their radios when they do not need to be active.

The mechanism of cluster-head election is totally distributed, so there is no need for topology knowledge. It is performed at defined time intervals. The time between two different elections is fixed and it is called a round. At each round not only does the cluster head change, but also clusters are totally rebuilt. So each round has a setup phase where clusters are built, followed by a phase where data transmission occurs. In order to maintain good performance for data transmission and moderate overheads, the data transmission phase has to be much longer than cluster creation. In detail, a round has four phases, as described below:

1. *Advertisement phase*: each node independently decides whether or not to become a cluster head for the current round. This choice depends on two main factors: the global percentage of cluster-head nodes, which is a predefined parameter for the network, and the number of rounds in which the node has already been a cluster head. More specifically, a generic node, n, makes the decision by generating a uniformly distributed random number in the interval [0–1] and comparing this number with a threshold $T(n)$. If the random number is below the threshold, it will become the cluster head for the current round, otherwise it will not. The exact value of the threshold is

$$T(n) = \begin{cases} \dfrac{p}{1 - p\left(r \cdot \bmod \frac{1}{p}\right)} & \text{if } n \in G \\ 0 & \text{otherwise} \end{cases} \tag{7.8}$$

   where
   p is the desired percentage of cluster heads
   r is the current round number
   G is the set of nodes that were not cluster heads during the last $1/p$ rounds

   Using this formula, each node will be a cluster head approximately once every $1/p$ rounds. Each self-elected cluster head broadcasts a ADV message to the non-cluster-head nodes using a fixed transmission power. For this transmission a simple CSMA MAC protocol can be used. Non-cluster-head nodes collect all the ADV messages received. When this phase ends, each node selects the cluster featuring the cluster head with the best signal strength as the one to join.
2. *Cluster set-up phase*: After a node has chosen the cluster to join, it has to communicate its choice to the relative cluster head. This communication can also use a simple CSMA MAC protocol.
3. *Schedule creation*: After the cluster heads receive messages from all the nodes that are to belong to its cluster, they can calculate a TDMA schedule for transmissions inside the cluster and broadcast it to all the cluster nodes. Furthermore, in order to avoid interferences with adjacent clusters, each cluster head also selects and communicates a CDMA code to be used during the transmission phase for the whole round.
4. *Data transmission*: The transmission schedule broadcast to the cluster nodes specifies the time slot in which each node has to transmit. Therefore, during the data phase, non-cluster-head nodes that have data to transmit wait for their time slot and then

send data during the allocated transmission time. As the radio channel is assumed to be symmetric, the transmitting power can be selected according to the signal strength of cluster-head packets. Nodes closer to the cluster head will thus transmit with lower power. Furthermore, as nodes can only transmit during their assigned time slot, they can shut down their radios during the time slots of other nodes, thus drastically reducing energy consumption. Cluster heads, on the other hand have to be active during the whole round, so they cannot shut down their radios.

Once data has been collected from every sensor in the cluster, the cluster head can run a data aggregation or specific signal composition algorithm in order to compress data into a single packet, and then it sends the packet right to the sink node with a high-energy transmission. Periodic data transmission will last until the end of the round, when a new election will start with the Advertisement phase.

While election can use a simple CSMA protocol, data transmission needs TDMA and CDMA at the same time in order to avoid collisions and interferences. TDMA is used to schedule transmissions inside the cluster, while CDMA reduces interference by nodes belonging to different clusters. Even if CDMA is not as efficient as the selection of a different radio frequency, it is simpler to implement than optimum channel selection. In fact, while the latter is shown to be NP-hard in [Sco96], a random CDMA code can solve the problem in a distributed way, albeit with some bandwidth overhead.

As the LEACH protocol performs power adjustment and duty-cycle adjustment and avoids collisions during data transmissions, its energy efficiency is very high. The benefit is more pronounced with protocols in which the duty cycle is not decreased, as low-power/sleep state consumption is usually some order of magnitude lower than the other states. Simulations in [Hei00] show that LEACH reduces the energy consumption of the network by a factor of 8 as compared with conventional routing protocols such as MTE, direct forwarding or static clustering. In addition, not only is the average energy consumption of nodes reduced, but also the energy is fairly distributed over the network, thanks to the rotation of the cluster heads. So the whole network lifetime will be significantly longer. However, there are also some drawbacks. First, data transmission is not continuous, as there are periodical interruptions to rebuild the clusters. Second, the ideal percentage of cluster heads varies with the number of WSN nodes or with the terrain size, so before a WSN can be efficiently deployed a simulation assessment may be necessary. Finally, the scalability of LEACH is limited by the single-hop approach adopted by cluster heads. In fact, although WSN nodes can often adjust the power level, the maximum range for small and energy-constrained motes cannot actually be very high. Thus, while the LEACH protocol is very energy-efficient, it cannot be used in large-scale WSNs where the sensing field exceeds the transmission range. In these cases a multi-hop approach is unavoidable.

7.5.2 LEACH Extensions and LEACH-Inspired Protocols

Several routing protocols which extend or improve the original LEACH protocol, or simply inspired by it, are proposed in the literature.

An extension of the LEACH protocol made by the same authors of the original protocol is presented in [Hei02], where an alternative selection algorithm for cluster heads is used. In fact, the basic election mechanism proposed in [Hei00] is totally probabilistic, and does not consider the energy consumed by the nodes when choosing whether or not to become a cluster head. But to improve the lifetime of the whole network, energy needs to be balanced. Hence, nodes with more energy should become cluster heads more frequently than nodes with less energy. This can be achieved by dynamically setting the probability of a node becoming a cluster head as a function of the current energy rather than as a function of the times the node has already been a cluster head. The proposed probability assignment is

$$P_i(t) = \min\left\{\frac{E_i(t)}{E_{\text{total}}(t)}k, i\right\} \tag{7.9}$$

where
k is the round
$E_i(t)$ is the current energy of node i
$E_{\text{total}}(t)$ is the sum of the current energy of all the nodes

Using this probability assignment, nodes with higher energy also have a greater probability of becoming cluster heads. However, this assignment requires knowledge of the energy of the other nodes in order to calculate $E_{\text{total}}(t)$, and acquisition of this knowledge itself would require a lot of energy. The solution proposed is that a node only collects the average energy for the cluster it belongs to and approximates the total energy by multiplying this value by the number of nodes N.

Another modification of the election phase is suggested in [Han02], where a deterministic factor is introduced in the probabilistic cluster-head selection. Here too the basic idea is to consider energy in cluster-head elections, in order to achieve a longer lifetime through clever cluster-head selection. The proposed approach is to modify the threshold equation $T(n)$, considering both the residual energy and the number of times a node has been a cluster head. The proposed threshold is

$$T(n)_{\text{new}} = \frac{p}{1 - p\left(r \cdot \bmod \frac{1}{p}\right)} \cdot \frac{E_{n\_\text{current}}}{E_{n\_\max}} \tag{7.10}$$

where E_{n_current} and $E_{n\_\max}$ are, respectively, the current and the initial energy levels. Using this new threshold, simulations show a lifetime increase between 20% and 30% over the original LEACH protocol. However, there is a nonnegligible drawback, as the probability of becoming a cluster head decreases as the energy for each node decreases, so the number of cluster heads will decrease until the network remains inactive, even if there are still nodes capable of transmitting to the sink node. Hence a further enhancement is proposed, with the threshold

$$T(n)_{\text{new}} = \frac{p}{1 - p\left(r \cdot \bmod \frac{1}{p}\right)} \left[\frac{E_{n\_\text{current}}}{E_{n\_\max}} + \left(r_s \operatorname{div} \frac{1}{p}\right)\left(1 - \frac{E_{n\_\text{current}}}{E_{n\_\max}}\right)\right] \tag{7.11}$$

where r_s is the number of consecutive rounds in which the node has not been a cluster head. Using this metric, after $1/p$ rounds during which a node is not a cluster head, the threshold reaches the same value it would have in the original protocol. In this way, clever cluster-head selection still increases the lifetime of the WSN, without causing network blockage.

7.5.2.1 LEACH-Centralized (LEACH-C)

The LEACH-C protocol described in [Hei02] proposes a further improvement for LEACH regarding cluster formation. In fact, the distributed protocol in LEACH does not offer guarantees on either the good placement or the correct percentage of cluster heads. So the basic idea of LEACH-C is to run a centralized clustering algorithm while maintaining the same data transmission phase. All the computations are performed at the BS, which has to know the position and energy of every node. In this way, the BS will only consider nodes with sufficient energy as possible candidates to become cluster heads. The objective of the clustering algorithm is to minimize the power consumption of the non-cluster-head nodes, so the optimal selection is mapped on a simulated annealing problem in which the total sum of the squared distance between the non-cluster-head nodes and the closest cluster head is minimized. Once the selection is performed, the BS broadcasts a message containing the cluster-head ID of each node. The rest of protocol is the same as LEACH.

7.5.2.2 Hybrid Energy-Efficient Distributed Clustering (HEED) Protocol

The HEED protocol presented in [You04] is an energy-efficient clustering protocol designed for WSNs that can be used in a generic cluster-based application or routing protocol. The aim of the

algorithm is to prolong the lifetime of a WSN. In HEED cluster-head selection is thus based on two different parameters. The primary parameter is the residual energy of each node, while the second parameter measures the intracluster communication cost, i.e., the number of neighbors. The idea is to use the primary parameter to perform a probabilistic choice of an initial set of cluster heads, and the second parameter to break ties between them, e.g., when a node is within the range of multiple cluster heads. This is an iterative algorithm in which nodes change their probability of becoming cluster-head $\mathrm{CH}_{\mathrm{prob}}$ at each iteration. This value is initially set to

$$\mathrm{CH}_{\mathrm{prob}} = C_{\mathrm{prob}} \times \frac{E_{\mathrm{residual}}}{E_{\mathrm{max}}} \tag{7.12}$$

where

C_{prob} is a constant that limits initial cluster-head candidatures (but has no direct impact on the final results)
E_{residual} is the residual energy of the node
E_{max} is its maximum (initial) energy

When nodes elect themselves to become cluster heads, they send an announcement message and then go into *tentative_CH* status if their $\mathrm{CH}_{\mathrm{prob}}$ is less than 1, or otherwise into *final_CH* status. Nodes that receive an announcement consider themselves covered. At each iteration, each uncovered node elects itself as a cluster head with a probability $\mathrm{CH}_{\mathrm{prob}}$, then every node doubles its $\mathrm{CH}_{\mathrm{prob}}$ value. Each node selects the least-cost candidate as its cluster head. Nodes that complete the HEED execution without selecting a cluster head in *final_CH* status consider themselves uncovered and elect themselves cluster heads with *final_CH* status. A *tentative_CH* can also become a non-cluster-head node if it finds a lower-cost cluster head.

This algorithm is proved to guarantee a bounded number of iterations before converging. The selection of cluster heads is energy-aware (so it selects better cluster heads than LEACH) and the clusters obtained are balanced. However, HEED requires multiple iterations, so the overhead and power consumption due to network management is greater than in LEACH. Nevertheless, simulation results show that the higher overhead is compensated for by the better cluster-head selection mechanism and the final result is an increased network lifetime.

7.5.2.3 Maximum Energy Cluster-Head (MECH) Protocol

The MECH [Cha06] protocol is an improvement on LEACH, in which the cluster setup phase takes into consideration both the radio coverage and the number of nodes currently belonging to the cluster. In this way, cluster placement over the network can be more uniform than with the distributed LEACH approach. The MECH protocol features localized coordination and control for cluster setup, while balancing and further reducing the overall dissipated energy. This can be achieved by addressing two shortcomings of LEACH. The first is the random election of cluster heads, which does not allow them to be evenly distributed over the field and thus be concentrated in a small area. In this case, high-power transmissions may be needed to reach far cluster heads. The second problem is that if a cluster head is far away from the sink node, direct transmission is not efficient (or not even possible). To overcome these drawbacks, a different cluster setup phase is proposed as well as a hierarchical intercluster routing scheme.

Like the LEACH protocol, MECH also partitions time into rounds, each one divided into different phases. During cluster initialization, each node broadcasts a hello message using a limited amount of power. Nodes count their neighbors, and when the number reaches the predefined number of nodes in a cluster *CN* (a global parameter of the WSN), the node broadcasts an ADV message that informs neighbors that it will be the cluster head for the next round. When a node receives an ADV, a back-off timer is started, and it does not send ADVs even if the *CN* value is reached. When this timer expires, the node selects the cluster head with the highest signal strength and sends a message to inform

it that it has joined the cluster. Setup and data transmission phases are analogous to LEACH, i.e., cluster heads calculate and communicate a TDMA schedule to all the nodes in the same cluster, so data transmission by each node can only occur in the assigned time slot. The only difference is that, besides data, nodes also transmit their current energy values to the cluster head. In this way, each cluster head knows which node has the highest energy inside its cluster and when the round ends this node can be directly elected as the cluster head for the next round. After the Data Transmission phase MECH introduces a Forwarding phase, in which a hierarchical relationship among clusters is defined. Thanks to this hierarchical relationship multi-hop data forwarding from a cluster head to the sink is now possible. LEACH, on the contrary, always uses direct transmission from cluster heads to the sink. The Forwarding phase uses two parameters to create the hierarchy: hop count from the sink node and energy. The phase is started by the sink node, which periodically broadcasts a message to all the cluster heads. Each cluster head updates the hop count and the minimum energy of the path, both contained in the message, and rebroadcasts the packet. At the same time, each cluster head maintains the ID of the best node for forwarding data towards the sink, that is, the node with the lowest hop count or, if several nodes have the lowest hop count, the one with the highest energy.

Simulations show that MECH improves on LEACH in terms of network lifetime thanks to the more balanced cluster setup, although it introduces more overhead. Thanks to the multi-hop forwarding, scalability for large-area WSNs is also improved. However, no QoS is addressed for multi-hop transmissions.

7.5.2.4 Threshold-Sensitive Energy-Efficient Sensor Network (TEEN) Protocol

The TEEN [Man01] is a hierarchical protocol based on LEACH. TEEN is also the first routing protocol explicitly targeted at reactive WSNs, where nodes sense their environment continuously, but data is not periodically transmitted, as only significant variations in the values of sensed parameters are reported. TEEN defines two thresholds, the hard threshold (H_T) and the soft threshold (S_T). These values determine when sensor data needs to be transmitted. More specifically, when the sensed value exceeds its H_T, the node switches on its transmitter and sends the sensed data and stores the value. Then, as long as the sensed value remains greater than H_T, the node will retransmit the sensed data and store its value every time the sensed value differs from the stored one by at least S_T. In this way, the number of data transmissions is greatly reduced, further improving both energy consumption and responsiveness.

TEEN also extends LEACH in terms of network architecture, providing hierarchical clustering. In LEACH and MECH nodes are grouped into two categories, cluster-head nodes and non-cluster-head nodes. In LEACH cluster heads communicate directly with the BS, while in MECH they can operate multi-hop transmissions based on hop count and energy. However, in both protocols only one-level cluster heads are defined. TEEN, on the other hand, uses a hierarchical clustering scheme, as depicted in Figure 7.4. Cluster nodes transmit data to their first-level cluster heads. These in turn forward data to the next-level cluster heads (e.g., second-level cluster heads), and so on, until the uppermost-level cluster heads, which transmit directly to the sink node, are reached.

As cluster formation and transmission is based on LEACH, but there are fewer data transmissions, TEEN provides more energy saving than LEACH. However, it only works for reactive networks, so it is not applicable when data has to be continuously updated. The problem of long distances between cluster heads and the sink is only partially resolved with hierarchical clustering. In fact, multi-hop forwarding is always toward higher-level cluster heads. By increasing the levels in the hierarchy, nodes have to cover a larger area and no guarantee is given that the distance from the sink is reduced. So the problem is not solved but only moved to the highest-level cluster heads. Hence, the scalability of TEEN in terms of geographical dimensions is low, as it is limited by the transmission range of the (highest-level cluster-head) nodes.

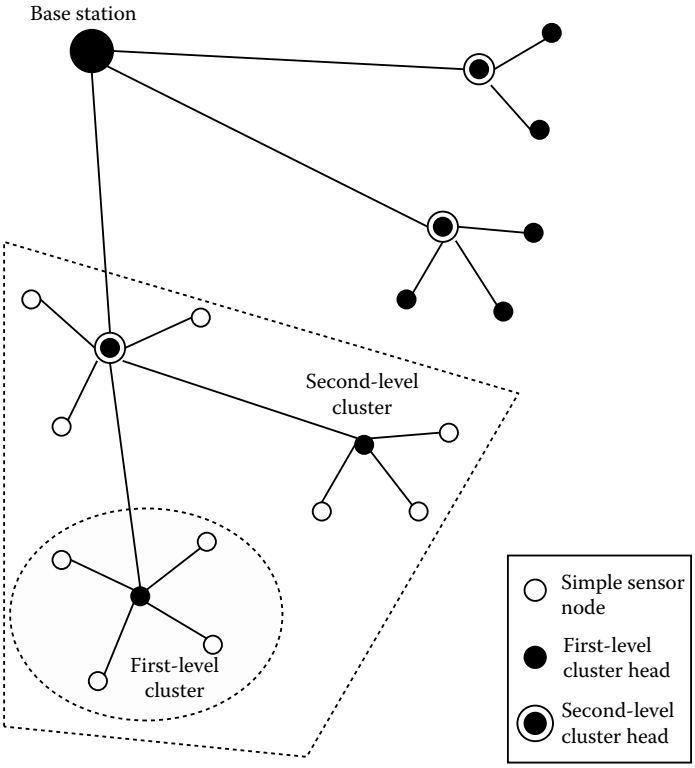


FIGURE 7.4 TEEN and APTEEN hierarchical clustering scheme.

7.5.2.5 Adaptive Threshold-Sensitive Energy-Efficient Sensor Network (APTEEN) Protocol

The authors of TEEN extend the protocol in [Man02] in order to efficiently manage different kinds of queries, implementing both proactive and reactive data transmission. This hybrid routing protocol, called the APTEEN) protocol, defines three different types of queries, i.e.,

- *Historical queries*, used for the analysis of historical data stored on the BS
- *One-time queries*, used to give a snapshot of the sensed environment at a defined time
- *Persistent queries*, used to monitor certain parameters for a defined time interval

Queries are received by WSN nodes and trigger interactions between nodes and the BS (i.e., the sink node). While in historical queries data is already stored in the BS, other queries may be stored or not depending on data criticality. Time-critical data can be fetched by the BS through periodical data transmissions in order to maintain values that are always up-to-date and can be directly sent when needed, while non-time-critical data can be transmitted on demand.

As the BS is able to transmit directly to any node, while nodes are energy-constrained, hierarchical addressing is used. Cluster formation and hierarchical addressing is the same as in TEEN. Data transmission combines TDMA for intracluster communications and CDMA to limit interference between different clusters. However, a different TDMA schedule for data transmission is proposed (Figure 7.5).

In APTEEN some nodes are assumed always to be in the listening state to receive queries. These nodes must always be active, and they should have a larger time slot than the others, as they may have to transmit both data and queries to their cluster head. Transmission by these nodes is scheduled

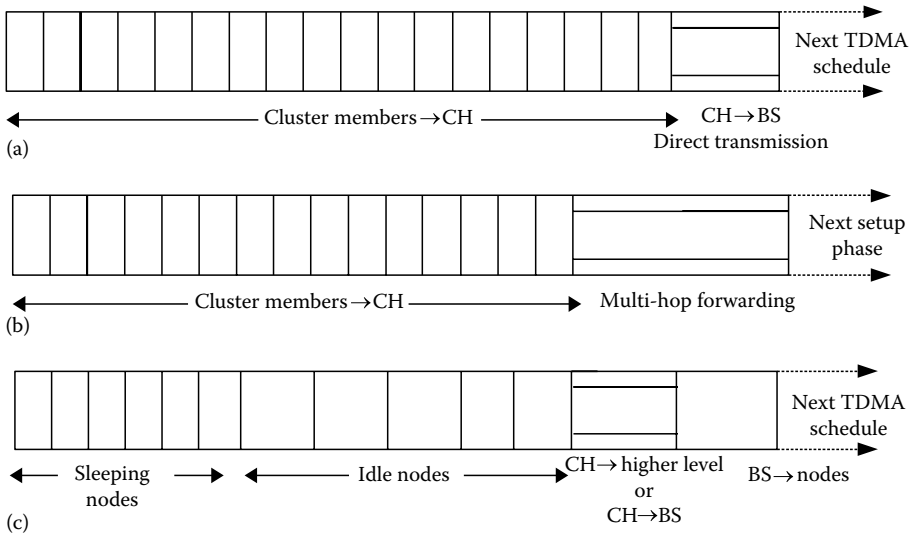


FIGURE 7.5 Super-frame comparisons between LEACH, MECH, and APTEEN. (a) LEACH transmission schedule. (b) MECH transmission and forwarding phases. (c) APTEEN transmission schedule.

after all the data transmissions of low duty-cycle nodes. As in TEEN, forwarding is always to the next-level cluster head, or directly to the sink node for the uppermost-level cluster head. But here the sink node can communicate directly with any node, and hence, in addition to a time slot for communication between cluster head and sink, the TDMA schedule also has a data time slot for transmissions between the sink node and other non-cluster-head nodes.

Regarding energy efficiency, APTEEN performs slightly worse than TEEN, as non-cluster-head nodes that are listening for incoming queries cannot go to sleep. This increased energy consumption is partially balanced by the reactive operating mode. There is a trade-off between energy consumption and response time, since the use of periodic transmissions can reduce the response time for the queries (the BS always has updated data, so it does not need to wait for the sensed data), but it also increases energy consumption. Nevertheless, APTEEN improves on LEACH, as it transmits data based on the threshold values, while LEACH transmits data all the time.

7.5.2.6 Power-Efficient Gathering in Sensor Information Systems (PEGASIS) Protocol

The PEGASIS protocol is a LEACH-inspired protocol proposed in [Lin02]. PEGASIS is not exactly a cluster-based protocol, as nodes are not explicitly grouped into clusters. PEGASIS is instead a chain-based approach, in which each node only communicates with a close neighbor and takes turns to transmit to the BS, thus reducing the amount of energy spent per round. This approach distributes the energy load evenly among the sensor nodes in the network.

The PEGASIS protocol is designed for a WSN containing homogeneous and energy-constrained nodes, with no mobility. The BS (sink) is fixed and far away from nodes. The radio model adopted here is the same as the one presented in [Hei00], i.e., the first-order radio model. Using this model, energy efficiency can be improved by minimizing the amount of direct transmissions to the sink node. This idea is common to the LEACH protocol, in which clustering is used to reduce both the duty cycle of the nodes and direct transmissions to the BS. A way in which energy efficiency can be further improved is to decrease the number of nodes that perform long-range direct transmissions. So the basic idea of PEGASIS is to have only one designated node that directly transmits to the BS in each round. This can be achieved with a linear chain-based approach, where sensor nodes form a

chain, in which gathered data moves from node to node, gets fused, and eventually a designated node will transmit it to the BS. As nodes take turns to transmit to the BS and transmissions are between close neighbors, the average energy spent by each node per round is reduced.

Data fusion, which is performed at each node (except end nodes), decreases the size of the aggregated data packet. The designated node that transmits to the BS (called the leader node) changes at each round, so that energy consumption is balanced over the network. Chain setup to minimize the total length is similar to the travelling salesman problem, which is known to be an NP-hard problem, so it is dealt with by a greedy algorithm run either by sensor nodes in a distributed way or by the BS in a centralized way and then broadcast to sensor nodes. In order to build the chain, nodes are assumed to have global knowledge of the WSN topology and location awareness. The proposed greedy algorithm for chain setup is started by the node furthest from the BS and then includes the neighbors in the chain. In this way, nodes which are distant from the sink are sure to have close neighbors, so that longer-distance transmission occurring when a node is the leader can be balanced by lower-power transmission when it is not.

PEGASIS divides time into rounds. While the chain remains the same, the leader role is deterministically rotated at each round, i.e., at round i the leader will be node (i mod N), where N is the total number of nodes. The leader node can start the transmission phase by transmitting a small token to a node (i.e., the first), which will fuse its neighbor data with its own to generate a single packet of the same length, which is then transmitted to its other neighbor (if it has two neighbors).

In the chain shown in Figure 7.6, the leader is node c2. The leader starts transmissions by sending a token to the first node in the chain, c0, which will pass its data to its neighbor c1. Node c1 fuses the node c0 data with its own and then transmits the packet to the leader. After node c2 passes the token to node c4, the latter transmits its data to node c3. Node c3 fuses the node c4 data with its own and then transmits it to the leader. Node c2 waits to receive data from both neighbors and then fuses its data with its neighbors' data. Finally, node c2 transmits one message to the BS. When a node dies, the chain needs to be rebuilt from scratch.

This protocol significantly reduces the overhead and the number of messages as compared with LEACH, so it outperforms LEACH in terms of network lifetime, at least with the model adopted. In fact, a LEACH network may have many cluster heads that need to transmit directly to the sink node, while in PEGASIS there is only one leader for the whole network. In addition, the number of messages received is also highly reduced, as in PEGASIS the leader only receives two messages, while in LEACH each cluster head receives n messages (where n is the number of nodes belonging to the cluster). This saves energy, as with the first-order radio model [Hei00] receiving a message is a costly operation. However, the model does not take idle power consumption into account, which could be as great as the receiving power [Rag02]. If idle and sleep power consumption is taken into account, LEACH will probably turn out to be more power-efficient than PEGASIS, as non-cluster-head nodes in LEACH can drastically reduce their duty cycle, while in PEGASIS nonleader nodes must always be active in order to receive the token. In addition, this approach causes a high end-to-end delay for the sensor data, because all the data from the whole WSN has to be gathered and forwarded before a sensed value can reach the sink. Delay grows with a growing network, and a single leader can also become a bottleneck. In the event of a fault, a large part of the network would not be able to forward data and the whole chain would have to be rebuilt, while in LEACH only faults in a cluster head cause data loss, and they are confined within the same cluster.

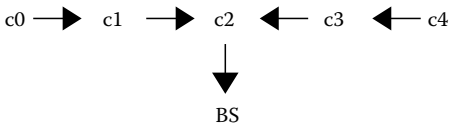


FIGURE 7.6 Chain-based forwarding in PEGASIS.

The PEGASIS protocol is extended in [Lin01], where not only energy efficiency but also delay is targeted. The authors investigate data gathering schemes that balance the energy and delay costs, as measured by the energy∗delay metric. Two different solutions are proposed, a chain-based binary scheme for sensor networks with CDMA nodes and a chain-based three-level scheme for sensor networks with non-CDMA nodes.

The two approaches share the same basic idea, i.e., achieving multiple parallel communications by different nodes. In order to improve energy and delay performance by simultaneous transmissions, interferences between different communications should be minimized. The first solution proposed to achieve multiple simultaneous communication with low interference is to use CDMA, so CDMA-capable sensor nodes are used. Each pair of nodes can use a distinct code to minimize interference with other transmissions and so a maximum degree of parallelism can be implemented. This means that in an N-node WSN, after the chain has been built as in PEGASIS, $N/2$ nodes can transmit at the same time with a different code (e.g., all the nodes in odd positions can simultaneously transmit to those in even positions). Then, at the second level of the hierarchy, only nodes that were receivers at the first level are considered (i.e., the nodes in even positions). So $N/4$ of this set containing $N/2$ nodes will be senders (i.e., once again the ones in even positions in the second-level set). Then the set of receiving nodes will be considered as the third-level set, and so on, until only one-node set remains. This is the leader node that has to perform direct transmission toward the sink. While in PEGASIS delay is a linear function of the number of WSN nodes N, using this approach delay is a logarithmic function.

However, CDMA may not be applicable for all sensor networks. The authors therefore investigate a second approach to achieve a minimal energy∗delay with non-CDMA nodes. In this case, in order to minimize interferences, only distant nodes should transmit at the same time. A three-level hierarchy for data gathering is therefore proposed, which allows simultaneous transmissions that are far apart so as to minimize interference while achieving a reasonable delay cost.

Simultaneous data transmissions are carefully scheduled taking the position of every node into consideration, so that at each level only the leaders of the previous level participate. The transmission schedule can be calculated once at the beginning, so that all nodes know where to send data in each communication round.

This approach is only based on experimental considerations and is not as efficient as the CDMA-based approach. However, it performs much better than LEACH and PEGASIS in terms of the energy∗delay metric. The energy model adopted here is the first-order radio model, as in LEACH and in PEGASIS, so idle (and sleep) power consumption is not taken into account. Even these extensions of the PEGASIS protocol do not directly decrease the duty cycle of nodes. Hence these results may change if the physical and MAC layers feature high idle power consumption.

7.5.2.7 Base Station-Controlled Dynamic Clustering (BCDPC) Protocol

The BCDPC presented in [Mur05] is a centralized cluster-based protocol that tries to combine the best features of LEACH and PEGASIS. BCDPC entrusts a high-energy BS with a large amount of energy supply with the task of setting up clusters and routing paths, performing randomized rotation of cluster heads and other energy-intensive tasks.

The key points in BCDCP are

- Formation of balanced clusters, where each cluster head serves an approximately equal number of member nodes to avoid cluster-head overload
- Uniform placement of cluster heads throughout the whole WSN, and adoption of cluster-head-to-cluster-head (CH-to-CH) routing to transfer the data to the BS

Like the PEGASIS protocol, BCDPC enables multi-hop forwarding of aggregated data toward a leader node, which directly transmits to the BS. However, here the forwarding is only CH-to-CH, through predefined paths that are computed and communicated by the BS.

Network functioning is divided into two phases: the setup phase and the data communication phase. During the former phase, cluster head selection, cluster setup, CH-to-CH path creation, and the transmission schedule have to be computed and then communicated to each node. Selection of the cluster heads is based on the residual energy of nodes. These values are sent in the transmissions of each node, so the BS always knows the energy status of each node. In order to identify a predefined number of cluster heads, N_{CH}, and to group nodes in a way that minimizes energy consumption in the data phase, an iterative cluster splitting algorithm is adopted, which first splits the network into two clusters and then iteratively splits each cluster into subclusters, until the desired number of cluster heads is obtained, in a way that maximizes the distance between the cluster heads of different clusters. Furthermore, the balanced clustering technique [Ghi02] is adopted to achieve a fair load distribution. The algorithm works in four steps. First, two nodes from the set of eligible cluster heads are chosen. Second, the remaining nodes are grouped with the closest of these two groups. Then, the two groups are balanced, so that their number of nodes is approximately the same. Finally, the set of eligible cluster heads is split into two sets according to the two groups created, and the next iteration can begin.

During the setup phase routing paths are also computed by the BS. These paths will be used to route data CH-to-CH toward the BS. Routing paths are selected by calculating the minimum spanning tree in terms of energy consumption for each cluster head [She99] and then randomly choosing the leader node, i.e., the cluster head that transmits directly to the BS.

The last task of the BS during the setup phase is to calculate the TDMA schedule used to transmit data.

During the data communication phase each node will use the assigned time slot to communicate with the cluster head. As in LEACH, CDMA is used to limit interference between multiple clusters. Data is gathered by the cluster head, aggregated, and then transmitted through the assigned CH-to-CH path.

As the number of long-range communications is reduced with respect to LEACH, energy efficiency is better. But it is also shown to be better than PEGASIS, as the (minimum energy) CH-to-CH path computed by the BS is more energy-efficient than the greedy algorithm used by PEGASIS, although in [Gua06] it is shown that the BCDPC routing scheme is not always energy-efficient. In addition (even if the authors do not explicitly address this feature), the use of a TDMA schedule enables simple duty-cycle reduction by putting nodes to sleep when they do not need to be active [Pot00]. However, this protocol is fully centralized, so scalability may be a problem for large WSNs, and the presence of a single leader as in PEGASIS may cause large delays.

7.5.3 Energy-Aware Routing Protocol in Cluster-Based Sensor Networks

A different approach to cluster-based WSNs was proposed in [You02], in which cluster heads are fixed, rather than elected in rotation, and called gateway nodes. Gateway nodes and sensor nodes are not the same kinds of nodes, i.e., only sensor nodes are battery-powered and thus energy-constrained. The main idea of this approach is to let gateway nodes be centralized network managers for nodes belonging to their clusters. Their role is not only to collect data from sensor nodes, as they also perform sensor organization and network management depending on the mission and residual energy of the node, so they have to set routes for data forwarding, monitor energy and latency throughout the cluster, and arbitrate medium access between sensor nodes.

This protocol assumes that both sensor nodes and gateways are in a fixed position (i.e., no mobility) and that each gateway is in the radio range of all the nodes in its cluster. Cluster formation is performed before the network starts and, as gateway nodes are not energy-constrained, there is no need to change them as in LEACH. However, the protocol itself does not describe the clustering algorithm adopted. Sensor nodes are assumed to be capable of independently shutting down their transceivers or their sensing circuitry and can operate in four different states.

In the sensing state, the sensing circuitry is active and a node periodically sends data to the gateway at a constant rate. In the relaying state, the sensing circuitry is off, but the transceiver is active to perform data forwarding. In the sensing–relaying state, a node performs both sensing and data forwarding so both the sensing circuitry and the transceiver must be active, while in the inactive state a node turns off both its sensing and communication circuitry. The state of each node is decided by the gateway depending on the node residual energy and the required performance. Gateway nodes make an estimation of the residual energy for each node in their cluster based on the number of transmissions. This is possible as the MAC protocol is a centralized TDMA, controlled by the gateway. However, to compensate for model inaccuracy, low-frequency periodical updates on the energy status are sent by sensor nodes.

Typical network functioning consists of two alternating phases: a data cycle, in which sensing nodes send data to their gateway, and a routing cycle, in which gateway nodes decide the optimal status and route assignments for each node and then directly transmit this information to nodes. It should be noted that while sensor nodes can forward data through multiple hops, gateways always perform direct transmissions. During the routing cycle, all the routing tables are computed by the gateway and then transmitted to each node. This is modelled as a path-optimization routing problem, where a least-cost path from each node to the gateway has to be found. The cost of a path is the sum of the costs of all the links traversed. A metric is provided that takes into account and balances several different parameters, including distance, residual energy, expected lifetime, overhead to switch a node from inactive to enabled status, sensing costs, maximum connection per relay, error rate, etc. In this way, the optimal route from each node to the gateway can be calculated using the Dijkstra algorithm [Dij59]. Besides routing tables, TDMA slot assignment also has to be scheduled by gateways and sent to nodes. Then the data cycle can start: nodes set their state and active ones start transmitting or forwarding during their time slots. At predefined times, all nodes have to switch on their receivers in order to receive new decisions from their gateway node. Rerouting may be triggered by application events or by the low battery charge of a node.

Energy efficiency in this protocol is achieved by shutting down sensing or communication circuitry when it is not needed. However, routing decisions are static for each data cycle, and may not change until a node runs out of charge. Even if the least-cost algorithm considers energy and the expected lifetime of networks, the choice is only performed once at the start of a routing cycle. The limit of static centralized routing is that, if no other changes occur in the network, rerouting will only be performed when some node has depleted its energy. Dynamically balancing network traffic could lead to a higher lifetime for the whole network.

Scalability is not affected by centralized routing, as several gateways can be deployed when the number of nodes per cluster increases. So scalability here could be better than in LEACH. However, this approach cannot be used where it is difficult to place multiple mains-powered gateway nodes.

This protocol was extended in [Akk03] with QoS management, in terms of end-to-end delay, which can efficiently handle best-effort traffic. This protocol will be dealt with in Section 7.7, where QoS-based EAR is addressed.

7.6 Location-Based Energy-Aware Routing Protocols

7.6.1 Minimum Energy Communication Network (MECN) Protocol

The MECN protocol [Rod99] is not specifically designed for sensor networks. However, the network model it proposes well suits WSNs. In fact, this is a distributed routing protocol where nodes are location-aware, i.e., equipped with a low-power global positioning system (GPS) [Sha97], and periodically transmit to a master-site node (the sink). Synchronous communications are used, so that nodes can sleep between subsequent communications, thus lowering their duty cycle and power consumption. MECN aims to be a self-configuring routing protocol that minimizes energy

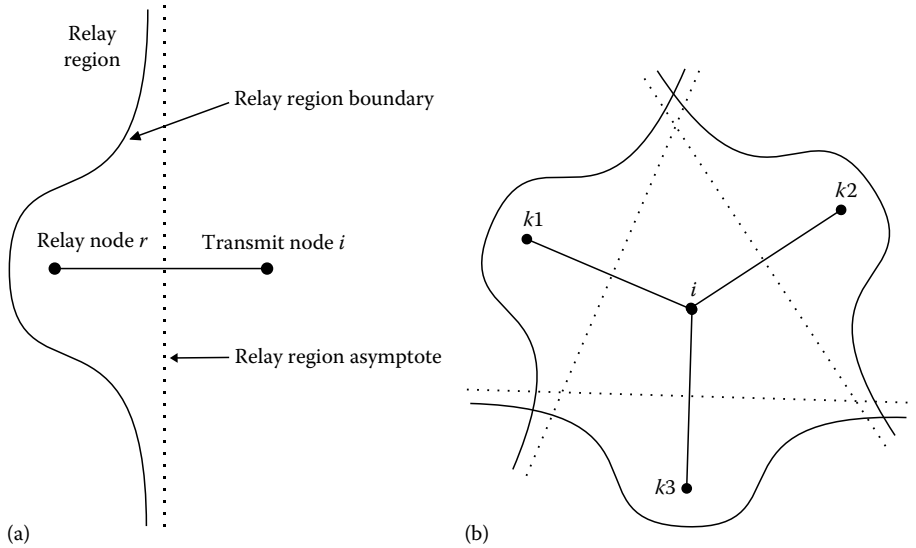


FIGURE 7.7 MECN regions. (a) Relay region of a transmit–relay node pair. (b) Representation of an enclosure. (Redrawn from Shahani, A.R., Schaeffer, D.K., and Lee, T.H., A 12 mW wide dynamic range CMOS front-end for a portable GPS receiver, in *Proceedings of IEEE International Solid-State Circuits Conference*, vol. 40, pp. 368–369, Feb. 1997.)

consumption for data forwarding by finding the minimum-power topology to route packets with. In MECN the authors observe that with the most common channel model used for RF systems [Rap96], the received power is proportional to $1/d^n$, where d is the distance and n is the path loss exponent (with $n \geq 2$ for outdoor propagation models). Using this model, the transmission power required is not proportional to the covering range, so relaying data between nodes can be more energy-efficient than direct transmission. But considering the sensor field as a two-dimensional plane, it is possible to calculate whether direct transmission is more or less expensive than relaying as a function of the receiver's position. Thus for each transmitter–relay node pair (i, r), a relay region can be defined as the region where forwarding through node r requires less energy than using direct transmission. A sample relay region is depicted in Figure 7.7a. For each node i, the enclosure is defined as the union of the relay regions (i, n), with $n \neq i$ (depicted in Figure 7.7b). This is the region around i beyond which it is not energy-efficient to perform forwarding, so it is not useful to take nodes into consideration for routing or to search for more neighbors. The main idea of this protocol is that, in order to find the minimum-energy path from a node to the sink, a very localized search can be performed that only considers nodes inside the enclosure.

The protocol is divided into two parts: a local search that finds the enclosure graph and minimum path construction. During the first phase, each node broadcasts a beacon containing its position and listens to beacons from neighbors. When it receives these messages, it computes the relay regions and only maintains nodes which do not lie in the relay regions of other neighbors. In this way the enclosure graph, which is the graph of communication links between all the nodes, is built. The second phase consists of finding optimal links in the enclosure graph. The algorithm adopted here is the distributed Bellman–Ford shortest path [Bel58] [For62], where power consumption is used as a cost metric.

Besides providing location information, GPS can also be used for synchronization purposes, so that nodes can synchronize their sleep and wake-up intervals. After a wake-up, a local search and minimum path construction have to be run in order to update optimal links. Hence the protocol is self-configuring and fault-tolerant, but requires a noticeable overhead. The protocol was extended

in [Li01] with a computationally simpler algorithm which builds smaller subnetworks, thus providing lower link maintenance costs and less overhead. However, while this protocol minimizes power consumption for data forwarding, it does not maximize the overall network lifetime, which is usually the main target in WSN.

7.6.2 Geographic and Energy-Aware Routing (GEAR) Protocol

The GEAR protocol, described in [Yu01], uses an energy-aware metric along with geographical information to efficiently disseminate data and queries across a WSN. Unlike other geographical protocols not specifically devised for sensor networks, such as the well-known greedy perimeter stateless routing (GPSR) protocol [Kar00], this protocol addresses the problem of forwarding data to each node inside a target region. This feature enables GEAR to support data-centric applications.

According to the GEAR protocol, each node has to know, besides its own location, the geographical position and residual energy of all its neighbor nodes. This can be accomplished through a low-frequency (and low-cost) hello message exchange. In addition, each query has to specify the target region, i.e., the area in which every node should receive the message. Each node maintains the learned costs, $h(N, R)$, for each neighbor–region pair where packets have to be forwarded. First, when there is no $h(N, R)$ entry for the neighbor N, this cost is computed as a function of the distance between N and the centroid of the region R and the energy consumed at node N. After the node has selected the next hop neighbor $N_{\min}$, the cost $h(N, R)$ is set to $h(N_{\min}, R)+C(N, N_{\min})$, where the last term is the communication cost from N to the selected neighbor $N_{\min}$. The estimated cost can subsequently be updated through feedback from the receiver node. In fact, after each packet is delivered, the learned cost is sent back to the last hop. Thus, if the destination is reached with n hops, after n subsequent packets to the same target the correct cost is propagated to the source node. Thanks to this mechanism it is possible to avoid holes simply by forwarding packets to nodes with minimum learned costs. Thus, the forwarding rule when the target region is not reached is always to send data to the neighbor N_i whose $h(N_i, R)$ is minimum. If there are no holes, the learned cost will only represent a combination of consumed energy and distance, so it will be equivalent to the estimated cost. In the presence of a hole, on the other hand, the updated learned cost will act as a "resistance" to following the path toward that hole.

Since the objective of this protocol is to disseminate queries inside the target region R, data forwarding does not end when a packet reaches that region, as data must be forwarded to every node inside R. To efficiently achieve this behavior two different mechanisms are proposed, i.e., the Recursive Geographic Forwarding or the Restricted Flooding algorithm.

Recursive Geographic Forwarding is used when node density is high. This is a recursive approach in which if a node N receives a packet destined to its region R, it splits R into four different subregions, and sends four copies of the packet, each targeted at one of these subregions. The recursive algorithm terminates when the current receiver is the only node inside the target region. This algorithm works well when node density is high, but with low densities it is inefficient and in some cases can never terminate and keeps routing uselessly around an empty target region before the packet's hop-count exceeds a certain bound. Thus when node density is low, the use of restricted flooding is suggested.

Restricted flooding exploits the broadcast medium of the wireless channel and only sends one broadcast message to all its neighbors, but every node in its transmission range receives this broadcast message whether it is an intended receiver or not.

This protocol achieves energy efficiency by means of the learned cost that takes residual energy into consideration along with geographical information about neighbors. By minimizing the energy cost between neighbors, an approximation of the lowest energy cost path is found. In order to improve network lifetime, the energy consumed so far is taken into account rather than the consumption of a single transmission. This protocol, compared with GSPR, features reduced energy consumption and a higher packet delivery ratio, at the expense of higher delay, as it takes longer paths in order to

balance energy consumption. However, this protocol assumes negligible energy consumption in the idle state, and every node should always be active. Thus, if idle power consumption is high, energy dissipation in the idle state will dominate the total energy consumption. However, the protocol does not have particular requirements for the MAC layer, so the authors suggest the use of a low-power MAC that puts itself to sleep when no activity is required.

7.7 Energy-Aware QoS-Enabled Routing Protocols

Energy-aware QoS-enabled routing protocols address the inherent conflict between the limited resources of each sensor node, particularly in terms of energy, and the need to achieve the desired QoS such as end-to-end real-time performance. In the following parts of this section some examples of EAR protocols able to provide some QoS for real-time data are described.

7.7.1 Energy-Aware QoS Routing Protocol for Wireless Sensor Networks

In [Akk03] an energy-aware QoS routing protocol for WSNs which can efficiently handle best-effort traffic is presented. The protocol finds QoS paths for real-time data with given end-to-end delay requirements. An analytic estimation of the end-to-end delay of a path is performed based on queuing theory. The protocol uses the estimated delays in order to select the delay-constrained path with the least cost. Here the cost of a link is a function of the distance between source and destination, energy, expected time, and error rate. In this way it is possible to find a path that meets delay requirements while optimizing energy and error rate. This protocol manages both real-time and non-real-time traffic, trying to balance low delay for real-time traffic with high throughput for non-real-time traffic. Each node is assumed to have two different queues, for real-time and non-real-time traffic, and a scheduler that determines the order in which data has to be sent. This order is based on a value r, which represents the amount of bandwidth to be used for real-time traffic, while $1–r$ is the ratio for non-real-time traffic. So an algorithm to calculate a proper r-value is proposed, which reduces the complexity of the problem by assuming an equal network-wide r-value for each link, so that a simple optimization problem can be formulated. The network-wide r-value guarantees a given service rate for real-time and non-real-time data on each link.

Simulation results in [Akk03] show that the protocol performs well with respect to both QoS metrics (i.e., throughput and average delay) and an energy-based metric (i.e., the average lifetime of a node). The results also show the impact of real-time data rate, buffer size, and packet drop probability on the performance of the protocol.

7.7.2 Real-Time Power-Aware Routing (RPAR) Protocol

In [Chi06] the RPAR protocol for WSNs is presented. The goal of this protocol is to obtain good soft real-time performance and minimize the deadline miss ratio, while also providing increased energy efficiency.

RPAR is based on empirical assessments of the impact of transmission power on the delivery velocity of each packet, defined as the total distance the packet travels divided by its end-to-end delay.

In [Chi06] it is shown that the impact of transmission power and one-hop distance on delivery velocity is significant. This is because the increased transmission power improves the quality of the wireless link, so the number of transmissions needed to deliver a packet is reduced. Moreover, the measurements performed show that the delivery velocity increases as the one-hop distance increases within a range, while it descends sharply when the one-hop distance exceeds the range due to link quality degradation. A higher transmission power increases such a drop-off range and allows for

higher delivery velocity. As a result, the authors infer that by controlling the transmission power it is possible to control the communication delay under light workloads.

However, the side effect of increasing the transmitting power is that the overall network capacity decreases due to increased contentions and interferences. Hence, the main idea of the RPAR protocol is to achieve a good trade-off between delay, energy consumption, and network capacity by dynamically adapting the nodes' transmitting power. That is, when deadlines are tight, the transmitting power is increased so that delay decreases, while when deadlines are loose, the transmitting power is lowered, so that energy consumption is reduced.

The RPAR protocol builds upon four components: a dynamic velocity assignment policy, a forwarding policy, a delay estimator, and a neighborhood manager.

The velocity assignment policy is used to transform the deadline requirements of a packet into velocity requirements. More specifically, given a source node S and a destination node D, the forwarding velocity required to meet the deadline is

$$v_{\text{req}}(S, D) = \frac{d(S, D)}{\text{slack}_{\text{rec}}(S) - (t_{\text{head}}(S) - t_{\text{rec}}(S))} \tag{7.13}$$

where

$d(S, D)$ is the Euclidean distance between S and D

$\text{slack}_{\text{rec}}(S)$ is the slack of the packet when it is received by S, i.e., the amount of time left before the deadline expires

$t_{\text{rec}}(S)$ and $t_{\text{head}}(S)$ are respectively the time instant when the packet is received and the time when it becomes the head of the transmission queue

If the velocity at each hop remains greater than the relevant $v_{\text{req}-}$ value, then the packet will meet the end-to-end deadline. Thus, the global problem of meeting an end-to-end deadline is mapped into a local problem of maintaining the velocity requirement of the single hop.

The delay estimator is assigned the task of assessing the one-hop delay for each forwarding choice maintained by the neighborhood manager. Each forwarding choice is a pair (N, p), where N is a neighbor node and p is the selected transmitting power. In general, the one-hop delay depends on the contention delay, $\text{delay}_{\text{cont}}(S)$, the transmission time of the packet and its acknowledgment, $\text{delay}_{\text{tran}}$, and the number of retransmissions, $R(S, (N, p))$. As a result, its value can be expressed as

$$\text{delay}(S, (N, p)) = (\text{delay}_{\text{cont}}(S) + \text{delay}_{\text{tran}}) \cdot R(S, (N, p)) \tag{7.14}$$

Since the transmission time is known from packet size and network bandwidth, the parameters that the delay estimator has to predict are contention delay and the number of retransmissions. A per-node estimation of delay and a per-forwarding-choice estimation of retransmissions are performed using the Jacobson algorithm [Jac88].

The forwarding policy chooses the most adequate forwarding choice for each packet, that is, the most efficient choice among those that reach the required velocity v_{prov}. So the velocity provided by each (N, p) pair has to be calculated before a forwarding decision can be made. The value can be calculated as

$$v_{\text{prov}}(S, D, (N, p)) = \frac{d(S, D) - d(N, D)}{\text{delay}(S, (N, p))} \tag{7.15}$$

where the term $d(S, D) - d(N, D)$ is the progress made toward the destination when node N is chosen as the next hop, while the denominator is the delay as provided by the delay estimator. In order to choose the most energy-efficient forwarding choice, the energy consumption of each choice

must also be computed. It can be expressed as

$$E(S, D, (N, p)) = E(p) \cdot R(S, (N, p)) \cdot \frac{d(S, D)}{d(S, D) - d(N, D)} \tag{7.16}$$

where

$E(p)$ is the energy spent to send the packet at power level p

$R(S, (N, p))$ represents the expected number of retransmissions per hop before S successfully delivers a packet to N when transmitting at power level p (as computed by the delay estimator)

the term $d(S, D)/(d(S, D) - d(N, D))$ estimates the expected number of hops to the destination

If the forwarding policy does not find a choice that meets the required velocity, the neighborhood manager comes into action, searching for new forwarding choices that could meet the requirement.

Two techniques are implemented to search for new forwarding choices: power adaptation and neighbor discovery. Power adaptation finds the most energy-efficient forwarding choice that still meets the velocity requirements. When velocity requirements cannot be met for a packet, the power adaptation scheme increases the transmission power to improve the velocity provided by neighbors already in the neighbor table. Conversely, when the velocity requirements are satisfied, the power adaptation scheme decreases the transmission power to improve the energy efficiency and network capacity.

Neighbor discovery is performed when no eligible forwarding choice is found with power adaptation. The aim is to identify nodes that meet the velocity requirement. Neighbor discovery starts by broadcasting a request to route (RTR) packets at power level p. A node N hears the RTR and replies. Upon receiving the reply, RPAR inserts the new forwarding choice (N, p) in its neighbor table.

As the number of all combinations (N, p) could be very high, the neighborhood manager only maintains the most frequently used entries to save space in memory-constrained devices.

This protocol addresses both real-time performance in terms of reduced deadline miss ratio and energy saving through efficient transmission power adaptation. Compared with SPEED-like protocols, both energy consumption and deadline miss ratio are reduced. However, in the presence of holes, Euclidean distance is a poor approximation of the path length, so QoS may be affected.

As explained in [Chi06], the RPAR power adaptation policy suffers from pathological behavior when a node is congested, as explained below. Due to high contention, there is a high collision probability, so a node needs a large number of retries before successfully transmitting a packet. As a consequence, RPAR increases the transmission power, thus worsening the situation. In [Chi06] some solutions to tackle this problem are outlined. One solution is to adopt MAC layer approaches enabling a node to distinguish between a packet being lost due to collisions or to poor link quality. Such feedback from the MAC layer would prevent RPAR from performing any useless and harmful increase in the transmission power. The second solution is the use of a congestion control protocol for WSNs able to detect a node congestion, thus enabling RPAR to stop increasing the transmission power. This solution would not only prevent the power control from worsening the congestion, but would also enable the congestion protocol to mitigate the problem.

Finally, this protocol does not handle sleep schedules, so energy consumption may still be high as compared to cluster-based approaches such as LEACH.

7.8 Topology Control Protocols for Energy-Efficient Routing

Topology control protocols are a slightly different approach to saving energy than standard routing protocols, as they do not directly operate data forwarding. These protocols run at a lower level of the

network stack, i.e., just under the network layer. Their objective is to improve the energy efficiency of routing protocols for wireless networks by coordinating the sleep transitions of nodes. Several routing protocols in fact try to enhance network lifetime by reducing the number of data transmissions or balancing the transmission power, but neglect idle power consumption. However, several measurements, e.g., in [Ka01] [Ste97], show that idle power dissipation should not be ignored, as it could be comparable to the transmitting or receiving power. Therefore, in order to optimize energy consumption, nodes should turn off their radios. Topology control protocols exploit redundancy in dense networks in order to put nodes to sleep while maintaining network connectivity. They can be applied to standard routing protocols for ad-hoc networks or for WSNs that do not directly handle sleep schedules. Although some of them are designed for wireless ad-hoc networks rather than WSNs, the typically high redundancy of sensor nodes and the need for maximum energy saving make WSNs perhaps the most suitable type of networks for taking advantage of these protocols. In the following sections, a few examples of topology control protocols are given.

7.8.1 Geographic-Adaptive Fidelity (GAF) Protocol

The GAF [Xu01] protocol, in order to put nodes into low-power sleep states without excessively increasing the packet loss rate, identifies groups of nodes that are "equivalent" in terms of routing cost and turn off unnecessary nodes. This is achieved by dividing the whole area into virtual grids, small enough that each node in a cell can hear each node from an adjacent cell. Nodes are location-aware, so each sensor obtains its coordinates on the virtual grid from location information. Nodes that belong to the same cell coordinate active and sleep periods, so that at least one node per cell is active and routing fidelity (which requires that in any cell at any one time there is at least one node able to perform routing [Xu00]) is maintained.

According to the GAF protocol, nodes can be in three different states: discovery, sleeping, and active. Transitions from one state to another are depicted in Figure 7.8.

In the discovery state nodes are active and exchange discovery messages in order to find nodes within the same cell. Then, after a discovery timeout T_d, a node enters the active state. Each node only stays in the active state for a defined time T_a, then moves to the discovery state again. A node in the discovery or active state can go to the sleep state if it finds an equivalent node with higher rank that handles routing. Finally, the sleep period is also limited, so that after a time T_s the node returns

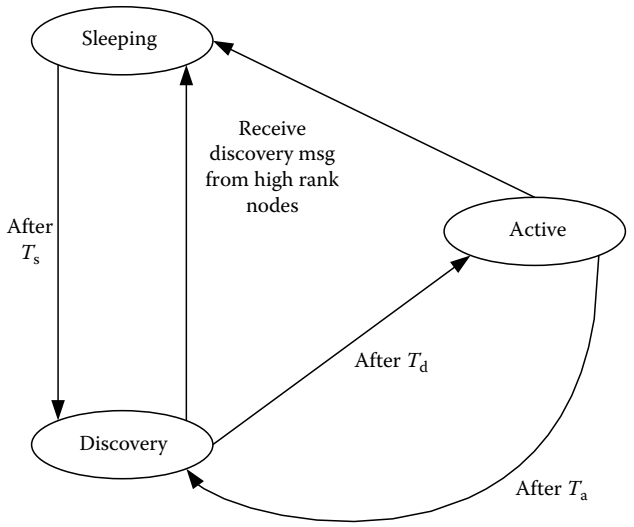


FIGURE 7.8 State transitions of the GAF protocol.

to the discovery state. In order to balance energy consumption and increase network lifetime, node ranking needs to be conveniently set. The value depends on the state and the residual energy of the nodes. Nodes in the active state always have a higher rank than those in the discovery state, while between nodes in the same state those with a longer expected lifetime have a higher rank.

GAF can run over any ad-hoc routing protocol, such as AODV [Perk99], dynamic source routing (DSR) [Joh96], TORA [Park97], destination-sequenced distance-vector (DSDV) [Perk94], and may be used for WSNs as well. Simulations show that with GAF there is no delay increase, while the consumed energy is highly reduced. The downside is that packet loss may slightly increase, as each time a previously active node goes to sleep there is a topology change the above routing protocol has to react to.

7.8.2 Span Protocol

In [Che02], another distributed coordination protocol for wireless ad-hoc networks, called Span, is presented. The objective of the Span protocol is to reduce energy consumption without significantly reducing network capacity or the connectivity of a multi-hop network. To achieve this, Span elects in rotation some coordinators that stay awake and actively perform multi-hop data forwarding, while the other nodes remain in power-saving mode and check whether they should become coordinators at regular intervals. Coordinators form a forwarding backbone that should provide as much capacity as the original network.

Each node makes periodic local decisions on whether to be a coordinator or not. Such decisions are based on a coordinator eligibility rule. If a node has two noncoordinator neighbors that cannot communicate with each other either directly or through other coordinators, then it will become a coordinator. In order to avoid contention in elections and to keep the number of coordinators small, nodes wait for a random delay period before sending their announcement message. Then they elect themselves as coordinators only if the eligibility rule still holds after the wait. A convenient selection scheme for the random period is proposed to keep the number of coordinators low and to achieve rotation. This takes two different factors into account, i.e., the number of additional pairs that will be connected if the node becomes a coordinator and its residual energy.

This protocol achieves significant energy saving while maintaining the performance of the upper-layer routing protocol almost unaltered. As compared to GAF [Xu01], it has an adaptive but less predictable number of forwarding nodes, as in GAF the grid is fixed and nodes inside the same cell are considered to be equivalent. Besides, Span does not require nodes to know their location. However, Span requires a modification to the lookup mechanism of the routing protocol, as only nodes in the active state have to be considered in choosing the next hop. Finally, the adoption of Span for WSNs could be constrained by the fact that it is designed for the IEEE 802.11 PHY and MAC protocol, e.g., it relies upon its ad-hoc power saving functions to buffer packets for sleeping nodes.

7.8.3 Sparse Topology and Energy Management (STEM) Protocol

The STEM protocol presented in [Sch02] is a topology control protocol specifically designed for WSNs. The assumption of STEM is that nodes in a WSN may spend most of the time only sensing the surrounding environment waiting for a target event to happen. Thus, unlike other topology management schemes that coordinate the activation of nodes during the transmission phase, STEM optimizes the energy efficiency of nodes during the monitoring state, i.e., when no one is sending data. STEM exploits the fact that, while waiting for events, the network capacity can be heavily reduced, thus resulting in energy savings.

So all nodes can be asleep when no data transmission is needed. On the other hand, it is important for nodes to be able to wake up neighbors when transmissions occur. The solution proposed by STEM is to keep the duty cycle of nodes very low unless data transmission starts. Nodes keep their radios

off most of the time, but periodically turn them on for a short time to hear whether some neighbor wants to communicate. A node that wants to communicate switches on its radio and starts sending beacons to the node it wants to wake up. Once the target node receives the beacon, it responds to the initiator node and data transmission starts. If data has to be forwarded further, the same operation is required for the next hop. In order to avoid collisions between data and wake-up beacons, the use of dual radio nodes is suggested, with two different frequency bands for the wake-up plane and the data plane.

In reactive WSNs, the STEM protocol can save more energy than other topology control protocols such as GAF or SPAN. However, it is not suitable for proactive WSNs, in which periodical updates have to be transmitted to the sink. Latency increases with the decreasing duty cycle of nodes, so a trade-off is needed between energy consumption and responsivity. Delay also increases linearly with the number of hops, so in large WSNs very high delays may be experienced.

Finally, it has to be highlighted that, rather than an alternative to other topology control protocols such as GAF or SPAN, the STEM protocol is orthogonal to them, so these approaches may coexist in the same network. As an example, the STEM-GAF combination is proposed in [Sch02]. This integration can reduce energy consumption even further.

7.9 Summary and Open Issues

Routing is one of the main research fields in the WSN area. As the main problem is to achieve durable networks despite the scarce energy resources sensor nodes are provided with, research into routing protocols for WSNs mainly focuses on energy-efficient techniques to disseminate data and/or queries. This chapter has presented some of the most relevant and best-known techniques for energy-efficient routing in the literature. Five different categories of routing protocols which adopt different approaches to achieving energy efficiency have been identified and characterized. The protocols surveyed have been described in the context of the category they belong to.

Although significant achievements have been obtained on the topic, there are still some open issues concerning energy-efficient routing in WSNs which deserve further investigation. One main issue is achieving a good trade-off between energy efficiency and QoS, and, in particular, between energy efficiency and soft real-time support, which requires bounded delays. As is known, duty-cycle reduction is the most effective way to reduce energy consumption, but this is also a cause of delay increase. For this reason, suitable cluster-based approaches which, while improving network lifetime, are able to achieve bounded delays, are especially sought.

Another promising line of research is network architectures able to exploit the advantages of different routing approaches. As an example, geographic routing approaches often achieve good performance in terms of delays [He03] [He05] [Chi06], but only a few of them are also energy-efficient. On the contrary, cluster-based routing algorithms are able to obtain significant energy savings thanks to the reduced duty cycles of nodes, but rarely address QoS. For this reason, hybrid protocols or frameworks, such as [Tos07] that combine the energy efficiency of a cluster-based topology control mechanism with the routing performance of a QoS-enabled routing layer are worth further investigation.

In addition to low energy consumption and bounded delay, routing techniques for WSNs should address application-dependent requirements, such as reliability, authentication, confidentiality, automatic set-up and reconfigurability. New research challenges are posed by sensor node mobility as well as by data gathering in the presence of multiple mobile sinks (e.g., laptops or PDAs).

Energy-efficient routing protocols for pervasive, large-scale WSNs and wearable sensor nodes are also sought. A promising development to achieve long-lived WSNs comes from sensor nodes able to apply energy harvesting techniques to capture energy from ambient sources. Some work already exists that tries to integrate sun power exploitation in the LEACH protocol [Thi04] [Isl07], but

more effort in this direction is expected. New challenges might also come from the exploitation, in a context-dependent way, of other energy sources, such as electromagnetic fields, fluid flows, and mechanical vibrations, or other techniques that try to harness the energy produced by the human body [Sta96], or the action of gravitational fields, e.g., [Hay91].

References

[Abi00] A.A. Abidi, G.J. Pottie, and W.J. Kaiser, Power-conscious design of wireless circuits and systems, *Proceedings of the IEEE*, 88(10), 1528–1545, Oct. 2000.

[Akk03] K. Akkaya and M. Younis, An energy-aware QoS routing protocol for wireless sensor networks, in *Proceedings of IEEE of the 23rd International Conference on Distributed Computing Systems*, pp. 710–715, 2003.

[Bel58] R. Bellman. On a routing problem, *Quarterly of Applied Mathematics*, 16(1), 87–90, 1958.

[Bra02] D. Braginsky and D. Estrin, Rumor algorithm for sensor networks, in *Proceedings of the 1st Workshop on Sensor Networks and Applications (WSNA)*, Atlanta, GA, Oct. 2002.

[Cha06] R.S. Chang and C.J. Kuo, An energy efficient routing mechanism for wireless sensor networks, in *Proceedings of the 20th International Conference on Advanced Information Networking and Application*, vol. 51, 2006.

[Cha99] J.-H. Chang and L. Tassiulas, Routing for maximum system lifetime in wireless ad-hoc networks, in *Proceedings of 37th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, Sept. 1999.

[Cha00] J.H. Chang and L. Tassiulas, Maximum lifetime routing in wireless sensor networks, in *Proceedings of Advanced Telecommunications and Information Distribution Research Program*, College Park, MD, 2000.

[Che02] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *ACM Wireless Networks Journal*, 8(5), 481–494, Sept. 2002.

[Chi06] O. Chipara, Z. He, Q. Chen, G. Xing, X. Wang, C. Lu, J. Stankovic, and T. Abdelzaher, Real-time power-aware routing in sensor networks, in *Proceedings of the 14th IEEE International Workshop on Quality of Service, 2006. IWQoS 2006*, pp. 83–92, Jun. 2006.

[Chu02] M. Chu, H. Haussecker, and F. Zhao, Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks. *International Journal of High Performance Computing Applications*, 16(3), 293–313, 2002.

[CT00] J.-H. Chang and L. Tassiulas, Energy conserving routing in wireless ad-hoc networks, in *Proceedings of IEEE INFOCOM '2000*, pp. 22–31, Israel, Mar. 2000.

[DaS01] J.L. da Silva Jr. et al., Design methodology for picoradio networks, in *Proceedings of the Design Automation and Test Conference (DATE)*, pp. 314–325, Germany, Mar. 2001.

[Dij59] E.W. Dijkstra, A note on two problems in connexion with graphs, *Numerische Mathematik*, 1:S. 269–271, 1959.

[Estr99] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, Next century challenges: Scalable coordination in sensor networks, *Proceedings of the 5th Annual ACM/IEEE International Conference Mobile Computing and Networking (MobiCom)*, pp. 263–270, Aug. 1999.

[Fel05] E. Felemban, C.-G. Lee, E. Ekici, R. Boder, and S. Vural, Probabilistic QoS guarantee in reliability and timeliness domains in wireless sensor networks, in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE INFOCOM 2005*, vol. 4, pp. 2646–2657, Mar. 13–17, 2005.

[Fel06] E. Felemban, C.-G. Lee, and E. Ekici, MMSPEED: Multipath multi-SPEED protocol for QoS guarantee of reliability and timeliness in wireless sensor networks, *IEEE Transactions on Mobile Computing*, 5(6), 738–754, Jun. 2006.

[For62] L.R. Ford and D.R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.

[Gad06] Y. Gadallah, A comparative study of routing strategies for wireless sensor networks: Are MANET protocols good fit? in *Ad-Hoc, Mobile, and Wireless Networks*, Springer, Berlin/ Heidelberg, pp. 5–18, 2006.

[Ghi02] S. Ghiasi et al., Optimal energy aware clustering in sensor network, *Sensors*, 2(7), 258–269, Molecular Diversity Preservation International (MDPI), Jul. 2002.

[Gua06] H. Guangyan, L. Xiaowei, and H. Jing, Energy-efficiency analysis of cluster-based routing protocols in wireless sensor networks, in *Proceedings of IEEE Aerospace Conference*, Mar. 2006.

[Han02] M.J. Handy, M. Haase, and D. Timmermann, Low energy adaptive clustering hierarchy with deterministic cluster-head selection, in *Proceedings of the IEEE International Conference on Mobile and Wireless Communications Networks*, pp. 368–372, Sweden, 2002.

[Hay91] M. Hayakawa, Electronic wristwatch with generator, U.S. Patent No. 5,001,685, Mar. 1991.

[He03] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, SPEED: A stateless protocol for real-time communication in sensor networks, in *Proceedings of the IEEE International Conference Distributed Computing Systems*, pp. 46–55, 2003.

[He05] T. He, J.A. Stankovic, T.F. Abdelzaher, and C. Lu, A spatiotemporal communication protocol for wireless sensor networks, *IEEE Transactions on Parallel and Distributed Systems*, 16(10), 995–1006, Oct. 2005.

[Hed88] S. Hedetniemi and A. Liestman, A survey of gossiping and broadcasting in communication networks, *IEEE Networks*, 18(4), 319–349, 1988.

[Hei00] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, in *Proceedings of the 33rd Hawaii International Conference on Systems Science*, vol. 8, pp. 3005–3014, 2000.

[Hei02] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, An application-specific protocol architecture for wireless microsensor networks, *IEEE Transactions on Wireless Communications*, 1(4), 660–670, 2002.

[Int00] C. Intanagonwiwat, R. Govindan, and D. Estrin, Directed diffusion: A scalable and robust communication paradigm for sensor networks, in *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 56–67, 2000.

[Isl07] J. Islam, M. Islam, and M.N. Islam, A-sLEACH: An Advanced Solar Aware Leach Protocol for Energy Efficient Routing in Wireless Sensor Networks, in *6th International Conference on Networking, 2007 (ICN '07)*, pp. 4-4, April 22–28, 2007.

[Jac88] V. Jacobson, Congestion avoidance and control, in *ACM SIGCOMM'88*, pp. 314–329, 1988.

[Jacq01] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, Optimized link state routing protocol for ad hoc networks, in *Proceedings of IEEE International Multi Topic Conference (INMIC)*, pp. 62–68, Pakistan, May 2001.

[Joh96] D.B. Johnson and D.A. Maltz, Dynamic source routing in ad hoc wireless networks, in *Mobile Computing* (Imielinski and Korth, eds.), vol. 353, Kluwer Academic Publishers, Hingham, MA, 1996.

[Kar00] B. Karp and H.T. Kung, GPSR: Greedy perimeter stateless routing for wireless networks, in *Proceedings of ACM/IEEE MobiCom*, pp. 243–254, Aug. 2000.

[Ka01] O. Kasten, Energy consumption. Available at http://www.inf.ethz.ch/~kasten/research/bathtub/energy_consumption.html, 2001.

[Kul99] J. Kulik, W. Rabiner, and H. Balakrishnan, Adaptive protocols for information dissemination in wireless sensor networks, in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 174–185, 1999.

[Li01] L. Li and J.Y. Halpern, Minimum-energy mobile wireless networks revisited, in *Proceedings of the IEEE International Conference on Communications, 2001. ICC 2001*, vol.1, pp. 278–283, Jun. 11–14, 2001.

[Lin97] C.R. Lin and M. Gerla, Adaptive clustering for mobile wireless networks, *IEEE Journal on Selected Areas in Communications*, 15(7), 1265–1275, Sept. 1997.

[Lin01] S. Lindsey, C. Raghavendra, and K. Sivalingam, Data gathering in sensor networks using the energy*delay metric, in *Proceedings of the 15th International Parallel and Distributed Processing Symposium*, pp. 2001–2008, Apr. 2001.

[Lin02] S. Lindsey and C.S. Raghavendra, PEGASIS: Power-efficient gathering in sensor information systems, in *IEEE Aerospace Conference Proceedings*, vol. 3, pp. 1125–1130, 2002.

[Man01] A. Manjeshwar and D. Agrawal, TEEN: A routing protocol for enhanced efficient in wireless sensor networks, in *Proceedings of the 15th International Parallel and Distributed Processing Symposium*, pp. 2009–2015, 2001.

[Man02] A. Manjeshwar and D. Agrawal, APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks, in *Proceedings of the 2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, Ft. Lauderdale, FL, Apr. 2002.

[Mur05] S.D. Muruganathan, D.C.F. Ma, R.I. Bhasin, and A.O. Fapojuwo, A centralized energy-efficient routing protocol for wireless sensor networks, *IEEE Communications Magazine*, 43, 8–13, Mar. 2005.

[Park97] V.D. Park and M.S. Corson, A highly adaptive distributed routing algorithm for mobile wireless networks, in *Proceedings of the IEEE INFOCOM*, pp. 1405–1413, Apr. 1997.

[Perk94] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers, in *Proceedings of the ACM SIGCOMM Computer Communication Review*, 24(4), 234–244, Oct. 1994.

[Perk99] C.E. Perkins and E.M. Royer, Ad-hoc on-demand distance vector routing, in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90–100, Feb. 1999.

[Pot00] G.J. Pottie and W.J. Kaiser, Wireless integrated network sensors, *Communications of the ACM*, 43(5), 51–58, May 2000.

[Rag02] V. Raghunathan, C. Schurgers, S. Park, and M.B. Srivastava, Energy-aware wireless microsensor networks, *IEEE Signal Processing Magazine*, 19, 40–50, 2002.

[Rap96] T.S. Rappaport, *Wireless Communications: Principles and Practice*, Prentice Hall, Englewood Cliffs, NJ, 1996.

[Rod99] T. Rodoplu and T.H. Meng, Minimum energy mobile wireless networks, *IEEE Journal on Selected Areas in Communications*, 17(8), 1333–1344, Aug. 1999.

[Sch02] C. Schurgers, V. Tsiatsis, and M.B. Srivastava, STEM: Topology management for energy efficient sensor networks, *IEEE Aerospace Conference Proceedings*, vol. 3, pp. 1099–1108, 2002.

[Sco96] K. Scott and N. Bambos, Routing and channel assignment for low power transmission in PCS, in *5th IEEE International Conference on Universal Personal Communications*, vol. 2, pp. 498–502, Sept. 1996.

[Sha02] R.C. Shah and J.M. Rabaey, Energy aware routing for low energy ad hoc sensor networks, in *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 1, pp. 350–355, Orlando, FL, Mar. 2002.

[Sha97] A.R. Shahani, D.K. Schaeffer, and T.H. Lee, A 12 mW wide dynamic range CMOS front-end for a portable GPS receiver, in *Proceedings of IEEE International Solid-State Circuits Conference*, vol. 40, pp. 368–369, Feb. 1997.

[She98] T. Shepard, A channel access scheme for large dense packet radio networks, in *Proceedings of ACM SIGCOMM*, pp. 219–230, Aug. 1998.

[She99] H. Shen, Finding the *k* most vital edges with respect to minimum spanning tree, *Acta Informatica*, 36(5), 405–424, Springer-Verlag, Sept. 1999.

[Soh99] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie, A self-organizing wireless sensor network, in *Proceedings of the 39th Annual Allerton Conference on Communication, Control, and Computing*, Urbana, IL, Oct. 1999.

[Soh00] K. Sohrabi, J. Gao, V. Ailawadhi, and G.J. Pottie, Protocols for self-organization of a wireless sensor network, *IEEE Personal Communications*, 7(5), 16–27, 2000.

[Sta96] T. Starner, Human powered wearable computing, *IBM Systems Journal*, 35(3 and 4), 618–629, 1996.

[Ste97] M. Stemm and R. Katz Measuring and reducing energy consumption of network interfaces in hand-held devices, in *Institute of Electronics, Information, and Communication Engineers (IEICE) Transactions on Communications*, vol. E80B (8), pp. 1125–1131, Aug. 1997.

[Sto99] I. Stojmenovic and X. Lin, GEDIR: Loop-free location based routing in wireless networks, in *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems*, Boston, MA, USA, pp. 1025–1028, Nov. 3–6, 1999.

[Thi04] T. Voigt, H. Ritter, J. Schiller, A. Dunkels, and J. Alonso, Solar-aware clustering in wireless sensor networks, in *Proceedings of the 9th IEEE Symposium on Computers and Communications*, vol. 1, pp. 238–243, Jun. 2004.

[Tos07] E. Toscano, O. Mirabella, and L. Lo Bello, An energy-efficient real-time communication framework for wireless sensor networks, in *International Workshop on Real-Time Networks (RTN 07)*, Pisa, Italy, Jul. 2007.

[Xu00] Y. Xu, J. Heidemann, and D. Estrin, Adaptive energy-conserving routing for multihop ad hoc networks, Tech. Rep. 527, USC/ISI, Oct. 2000.

[Xu01] Y. Xu, J. Heidemann, and D. Estrin, Geography-informed energy conservation for ad hoc routing, in *Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 70–84, Rome, Italy, Jul. 2001.

[Yao02] Y. Yao and J. Gehrke, The Cougar approach to in-network query processing in sensor networks, *ACM SIGMOD Record*, 31(3), 9–18, 2002.

[Ye01] F. Ye, A. Chen, S. Lu, and L. Zhang, A scalable solution to minimum cost forwarding in large sensor network, in *Proceedings of 20th IEEE International Conference on Computer Communications and Networks*, pp. 304–309, 2001.

[Ye02] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, A two-tier data dissemination model for large-scale wireless sensor networks, in *Proceedings of the 8th ACM International Conference on Mobile Computing and Networking (MobiCom 2002)*, pp. 148–159, Atlanta, GA, Sept. 2002.

[You02] M. Younis, M. Youssef, and K. Arisha, Energy-aware routing in cluster-based sensor networks, in *International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 129–136, Oct. 2002.

[You04] O. Younis and S. Fahmy, Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach, in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE INFOCOM*, vol. 1, pp. 629–640, Mar. 2004.

[Yu01] Y. Yu, R. Govindan, and D. Estrin, Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks, UCLA Computer Science Department Technical Report UCLA/CSD-TR-01-0023, May 2001.

[Zha07] L. Zhao, B. Kan, Y. Xu, and X. Li, FT-SPEED: A fault-tolerant, real-time routing protocol for wireless sensor networks, in *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing, WiCom 2007*, pp. 2531–2534, Sept. 21–25, 2007.

8

Energy-Efficient MAC Protocols for Wireless Sensor Networks

8.1 Design Issues for MAC Protocols for WSNs **8**-1
Causes of Energy Waste in WSNs • Performance Metrics for MAC Protocols Used in WSNs
8.2 Overview on Energy-Efficient MAC Protocols for WSNs **8**-3
Power-Aware Multi-Access Protocol with Signaling • Sensor MAC • Timeout-MAC • Data-Gathering MAC • Berkeley MAC • WiseMAC • Power-Efficient and Delay-Aware MAC for Sensor Networks • Traffic-Adaptive Medium Access • Flow-Aware Medium Access • Sift • Lightweight Medium Access • Z-MAC • Pattern MAC • Crankshaft • Correlation-Based Collaborative MAC
8.3 Mobility Support in WSNs **8**-16
Mobility-Aware MAC for Sensor Networks • Mobility-Adaptive Collision-Free MAC • Mobility-Adaptive Hybrid MAC
8.4 Multichannel Protocols for WSNs **8**-18
Multifrequency Media Access Control for WSNs • Multichannel LMAC • Multichannel MAC
8.5 Summary and Open Issues **8**-20
References **8**-21

Lucia Lo Bello
University of Catania

Mario Collotta
University of Catania

Emanuele Toscano
University of Catania

8.1 Design Issues for MAC Protocols for WSNs

The primary objective in wireless sensor networks (WSNs) is to prolong the network lifetime as much as possible. As sensor nodes are typically battery-operated and WSNs are usually deployed in hostile or hard-to-reach environments, where it is difficult or impractical for human operators to access sensor nodes and recharge/replace them, energy efficiency is the primary requirement which a good medium access control (MAC) protocol for WSNs has to meet.

In addition to energy efficiency, other desirable properties for MAC protocols for WSNs are small code size and memory requirements, which derive from the limited resources sensor nodes are generally equipped with, and scalability, as WSNs typically consist of a large number of nodes spread over a broad area. Another requirement for MAC protocols for WSNs is adaptability to changes in network size, node density, and topology, which may occur due to limited nodes' lifetime, addition of new nodes and varying interference.

Traditional performance parameters such as throughput and bandwidth utilization play a secondary role in WSNs MAC protocols as compared to energy efficiency and scalability. Similar considerations hold for latency, so MAC protocols for WSNs may trade latency for energy efficiency. However, in the cases in which latency is an issue (e.g., in real-time WSNs or in wireless sensor/actor networks, WSANs), solutions to bound end-to-end latency are needed. Some works provide specific frameworks implementing solutions in which the MAC and Routing layers cooperate to achieve both energy efficiency and real-time support, e.g., the Real-Time Power Aware Routing protocol [Chi06]. Finally, fairness among sensor nodes is generally not an issue, since all of them cooperate to accomplish a common task. It may happen that, at any given time, one node has more data to send than other nodes. As long as application performance is not degraded, allowing such a node to transmit more than other nodes is acceptable.

While designing a MAC protocol for WSNs the traffic pattern to be dealt with has to be taken into account. The communication patterns typically found in WSNs can be classified as broadcast, convergecast, local gossip, and multicast [Kul04] [Dem06].

- *Broadcast*: A base station (or a sink node) sends data to all the sensor nodes in the WSN.
- *Local gossip*: Neighboring nodes communicate with each other locally, following the detection of an event.
- *Convergecast*: A group of sensors communicate what they perceived to a specific sensor (e.g., the sink, a cluster-head, etc.).
- *Multicast*: A sensor sends a message to a specific subset of sensors. An example of multicast communication can be found in cluster-based protocols where cluster-heads may need to send a message to a subset of their neighbors, i.e., the members of their cluster only.

As the primary goal of MAC protocols for WSNs is energy efficiency, in Section 8.1.1 the main causes of energy waste in WSNs are identified and discussed.

8.1.1 Causes of Energy Waste in WSNs

MAC protocols for WSNs have to cope with the main causes of energy waste in wireless networks that are collisions, overhearing, idle listening, protocol overhead, and overemitting.

- *Collisions*: When a transmitted packet is corrupted due to a collision, it has to be discarded and, after a random back-off procedure is run, retransmissions occur. These activities increase energy consumption. Moreover, when the workload approaches the available bandwidth, sensing for a clear channel consumes a significant amount of energy, although the number of successful transmissions will be extremely low.
- O*verhearing*: It occurs because the radio channel is a shared medium and thus a node may receive packets that are not destined for it.
- *Protocol overhead*: It is due to sending and receiving control packets (such as, ACKs, RTS/CTS, etc.) and MAC headers. Such an overhead may be significant, as in typical WSN applications small data are exchanged, thanks also to the in-network processing that allows for condensing multiple raw sensor readings into short information.
- *Idle listening*: Listening to receive possible traffic that is not sent. This happens because a node does not know when it will be the recipient of a message from one of its neighbors, so it must keep its receiver switched on at all times.
- *Overemitting*: It occurs when a node sends a message and the destination node is not ready to receive (e.g., as the node is in sleep state).

Idle listening represents the major source of energy waste in WSNs. A possible solution to reduce it is the low-power listening (LPL) [Hil02]. Such an approach operates at the PHY layer introducing a duty cycle. The basic idea is to shift the cost from the receiver to the transmitter by increasing the length of the preamble. This allows the receiver to periodically turn on the radio to sample for incoming data. If the receiver detects a preamble, it will continue listening until the start symbol arrives and the message can be properly received, otherwise it will turn off the radio and go to sleep until the next sample. Energy saving with LPL entails a slight degradation in both latency and throughput. LPL can be applied to any MAC protocol provided that the radio-switching time is short.

8.1.2 Performance Metrics for MAC Protocols Used in WSNs

The objective of MAC protocols for WSNs is to efficiently regulate the medium access, in terms of both performance and energy efficiency, so a relevant metric is the channel access success probability provided to the upper layers. Other metrics that are widely used to assess the effectiveness of a MAC protocol are throughput and delivery ratio. However, in some WSN applications not all the traffic has to be necessarily delivered in order to achieve reliable event detection, and redundant frames may be dropped for the sake of the network performance and energy efficiency. As a result, for such cases neither the throughput nor the delivery ratio is suitable performance metrics. In [Vur06] the goodput is used, which is defined as the ratio of the packets received by the sink and the number of packets that are actually generated by the nodes, thus taking into account early discarding of redundant data as well. Another parameter of interest in these contexts is the event detection reliability expressed, for example, in terms of distortion.

Among the metrics which are representative of the energy consumption or energy saving, the most widely adopted are the average energy consumption per node, used, for example, in [Dam03] [Hal07] [Vur06], and the average power consumption of nodes [Enz04]. As the energy consumption is directly related with the duty cycle of nodes or with the average sleep time, these parameters are also suitable metrics for an energy-efficient MAC protocol. In [Raj06], the percentage of sleep time is used together with the average length of the sleep interval. As the sleep and wake-up transitions of the radio are both time- and energy-consuming, the average length of the sleep interval is useful to estimate the amount of radio-mode switching and so the actual energy saving. In MAC protocols which provide for dynamic or adaptive nodes' duty cycles (e.g., T-MAC), a metric that reflects the energy consumption of every node is the distribution of active times, used e.g., in [Hal05].

There are WSNs in which timeliness is an important requirement. In these networks, the channel access delay is also an important metric. However, as this delay depends on the nodes' duty cycles, it usually clashes with energy consumption. As a result, it is usually more interesting to evaluate the trade-off between energy consumption and latency as, for example, in [Pol04], rather than the delay value alone.

8.2 Overview on Energy-Efficient MAC Protocols for WSNs

MAC protocols for WSNs can be classified into two basic classes: contention-based or schedule-based. In contention-based protocols, the sender has to continuously sense the channel to assess when it is idle, collisions may occur and a back-off procedure has to be run before retransmitting after a collision. As said before, all these activities are energy-consuming. On the contrary, in schedule-based protocols, data transfers are scheduled in advance and the recipients know exactly when to switch on the radio and when they can safely go to sleep.

Both these protocol classes have their pros and cons. On the one hand, contention-based protocols are energy-consuming due to collisions, idle listening, and overhearing. However, they have some valuable benefits, such as, low implementation complexity, scalability, and flexibility

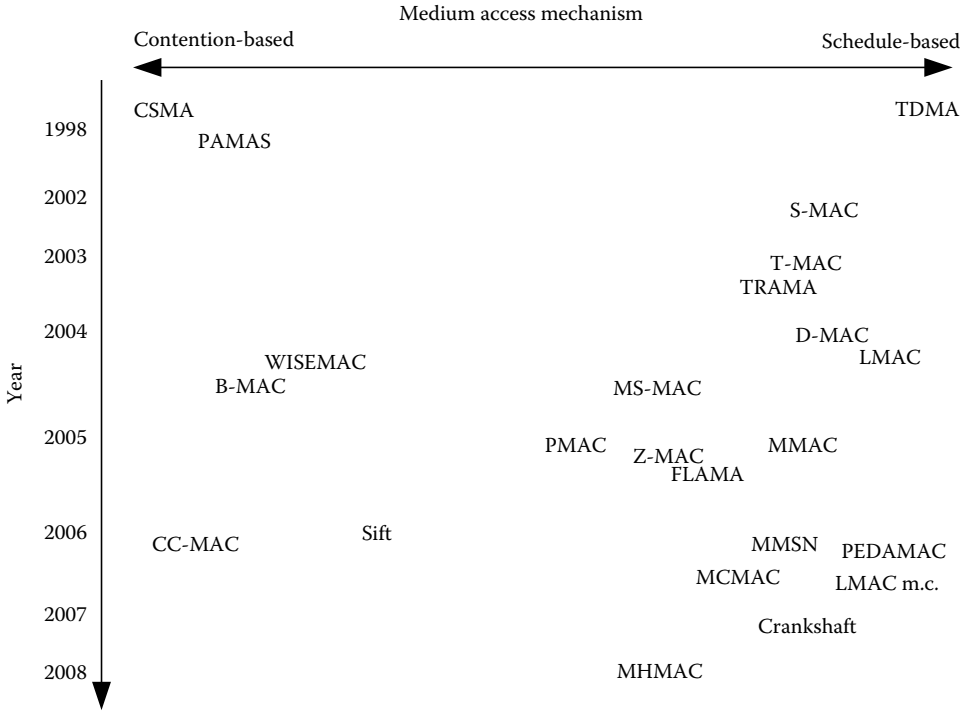


FIGURE 8.1 Classification of MAC protocols for WSNs.

to deal with mobile nodes and network changes. On the other hand, schedule-based protocols are energy-efficient, as they have a duty cycle built-in and are collision-free, but they are more complex to implement. The need for broadcasting a schedule in advance requires some coordination between the nodes, and also limits scalability and flexibility to handle mobile nodes and to deal with varying topology and network size. Moreover, maintaining a list of neighbors' schedules requires a significant amount of memory.

From what was said above, it is clear that schedule-based protocols offer some advantages which contention-based ones do not have, and vice-versa. In the WSN literature therefore we hardly find MAC protocols that strictly belong to one of these categories, e.g., a pure carrier-sense multiple access (CSMA) or a pure time division multiple access (TDMA) protocol. For the sake of energy efficiency and performance improvement, the majority of protocols try to combine the positive features of the two classes, so in the range from pure contention-based to pure schedule-based protocols, they can be placed in an intermediate region between two extremes, respectively CSMA and TDMA, as shown in Figure 8.1.

In the following, notable example of MAC protocols belonging to the different classes will be addressed.

8.2.1 Power-Aware Multi-Access Protocol with Signaling

The Power-Aware Multi-Access Protocol with Signaling (PAMAS) [Sin98] is based on the idea of switching off the radio of nodes when they are likely to overhear transmissions, i.e.,

- If a node has no packets to transmit and a neighbor begins transmitting or
- If at least one neighbor of a node is transmitting and another one is receiving, as in this case the node cannot transmit or receive a packet (even if it has a packet to transmit).

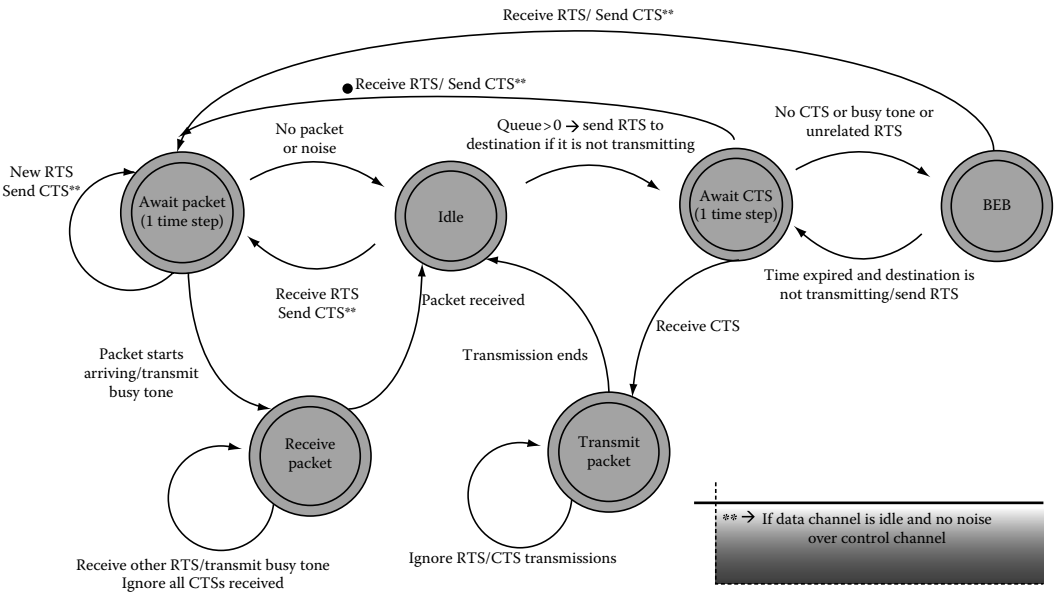

Receive RTS/ Send CTS**
Receive RTS/ Send CTS**
No packet or noise
Queue>0 → send RTS to destination if it is not transmitting
No CTS or busy tone or unrelated RTS
New RTS Send CTS**
Await packet (1 time step)
Idle
Await CTS (1 time step)
BEB
Receive RTS Send CTS**
Packet received
Time expired and destination is not transmitting/send RTS
Receive CTS
Packet starts arriving/transmit busy tone
Transmission ends
Receive packet
Transmit packet
** → If data channel is idle and no noise over control channel
Receive other RTS/transmit busy tone Ignore all CTSs received
Ignore RTS/CTS transmissions

FIGURE 8.2 PAMAS protocol. (Redrawn from Singh, S. and Raghavendra, C.S., *Comput. Commun. Rev.*, 28(3), 5, 1998.)

PAMAS adopts an out-of-band signaling scheme with two separate channels, one used for signaling and one for data transmission. An RTS–CTS message exchange takes place over the signaling channel and any node hearing the signaling destined to another node is able to determine when and for how long it can turn off its radio. A diagram that resumes the protocol operations is shown in Figure 8.2.

A node is aware of the fact that a neighbor is transmitting, because it can hear the transmission over the data channel. Likewise, a node which has data to transmit knows if one or more of its neighbors is receiving, as the receivers transmit a busy tone over the signaling channel when they begin receiving a packet and in response to RTS transmissions. For these reasons, any node in the system can independently decide when to switch off.

To determine for how long the node should be switched off, there are two cases. If a packet transmission in the neighborhood begins while the node is still switched on, the node knows the duration of that transmission thanks to the RTS/CTS exchange. If there is an on-going transmission when the node is back and switches on the radio, the node will use a mechanism based on probe packets to assess the remaining transmission time. The probe mechanism enables nodes to estimate the length of time that the radio can be switched off. Although the conservative approach proposed in [Sin98] might underestimate the length of this period, thus resulting in suboptimal power saving, it ensures that the delay/throughput of PAMAS remains unchanged.

In [Sin98] simulation results proved that PAMAS achieves significant energy saving. However, there are two drawbacks. First, collisions may still occur between probe messages or RTS/CTS packets. Second, out-of-band signaling requires a complex radio which may be unfeasible to be embedded in low-cost sensor nodes.

8.2.2 Sensor MAC

The Sensor MAC (S-MAC) protocol proposed in [Ye02] [Ye04] represents a milestone among energy-efficient MAC protocols for WSNs, as it provides the insights upon which many other protocols were

built. S-MAC aim is to reduce energy consumption from all the sources that cause energy waste, i.e., idle listening, collisions, overhearing, and protocol overhead. The S-MAC protocol consists of three major components:

- Periodic listen and sleep scheme
- Collision and overhearing avoidance mechanism
- Message passing

The idea behind periodic listen and sleep state is that in many WSN applications nodes may remain idle for a long time waiting for the occurrence of an event, so they do not have to be listening all the time. The S-MAC protocol therefore exploits this feature by letting nodes go into periodic sleep mode. While in the sleep mode, a node completely switches off its radio and sets a timer which is used to turn it on afterwards. This significantly reduces the actual listening time and thus the relevant energy consumption.

An entire cycle of listen and sleep is called a frame. The listening interval is normally fixed according to the radio bandwidth and the contention window size. The duty cycle is given by the ratio of the listening interval to the frame length. The sleep interval may change depending on the application requirements. All the nodes have the same values as listen and sleep intervals and they are free to choose their own listen/sleep schedules. However, to keep the protocol efficient while decreasing the overhead, neighboring nodes coordinate their sleep schedules, so that they listen at the same time and go to sleep at the same time, rather than randomly sleep on their own.

Nodes exchange their listen/sleep schedules by broadcasting them to all the immediate neighbors by periodic SYNC packets (including the address of the sender and the time of its next sleep) and each node maintains a schedule table that stores the schedules of all its known neighbors. Although this coordinated approach requires periodic clock synchronization among neighboring nodes, such a synchronization is significantly looser than the one required by TDMA schemes with very short time-slots, as in S-MAC the listening period is significantly longer than typical clock drift rates.

If multiple neighbors want to send DATA packets to a node, they go through a contention mechanism similar to the one used in the IEEE 802.11 standard, including both virtual and physical carrier sense and RTS/CTS (request to send/clear to send) exchange. The node whose RTS packet is received first wins the contention and the receiver will reply with a CTS packet. Once their data transmission starts, they do not follow their sleep schedules until the end of the transmission.

In order for a node to receive both SYNC packets and DATA packets, its listening interval is divided into two parts. The first part is for receiving SYNC packets, while the second one is for receiving DATA packets. The S-MAC protocol tries to avoid overhearing by allowing interfering nodes to go to sleep after they hear an RTS or CTS packet until the current transmission is over.

Since DATA packets are normally much longer than control packets, this approach prevents neighboring nodes from overhearing long DATA packets and the following ACKs. Each node maintains a network allocation vector (NAV) and an associated timer to keep track of the activity in its neighborhood. When a node receives a packet destined to other nodes, it updates its NAV by the duration field in the packet. As nonzero NAV value indicates that there is an active transmission in the neighborhood, a node with a nonzero NAV has to sleep to avoid overhearing. The NAV value decrements when the associated timer expires and a node can wake-up when its NAV reaches zero.

The message passing scheme proposed in S-MAC also aims to keep nodes in the sleep state as long as possible to reduce their switching overhead. To this aim, all DATA packets have a duration field, which represents the time needed for transmitting all the remaining data fragments and ACK packets. If a neighboring node hears a RTS or CTS packet, it will go to sleep for the time needed to transmit all the fragments.

Although S-MAC proposes the concept of virtual clusters that consist of synchronized nodes, there is no real clustering. This means that S-MAC does not define hierarchical relations between nodes,

instead it is a flat peer-to-peer protocol. S-MAC is an energy-efficient protocol simple to implement, but it has a drawback, i.e., the latency increases due to the periodic sleep of the receiver, which makes the sender wait for the receiver to wake-up before it can send out data. Such a time is called a sleep delay. This delay is particularly significant in multi-hop networks, as each hop introduces a sleep schedule. The latency requirement of the application constraints the maximum sleep time. The adaptive listening technique proposed in [Ye04] improves the sleep delay and thus the overall latency. In this approach, a node that overhears the transmission of a neighbor wakes up for a short time at the end of the transmission (which is known from the duration field of the transmitted packet). In this way, if the node is the intended next-hop recipient, its neighbor can immediately send data to it. Otherwise, it will go back to sleep until its next scheduled listening time.

S-MAC enables low-duty-cycle operation in a multi-hop network, achieving energy efficiency and also good scalability and collision avoidance through a combination of schedule-based and contention-based schemes.

However, the fixed duty cycle in S-MAC represents a limitation, as the active time mainly depends on the message rate, which will usually vary over time and space in a WSN. If important messages are not to be missed in any case, the active time has to be conservatively sized according to the worst-case, i.e., in such a way that the highest expected load can be handled. However, if the load is lower than that, which is often the case in WSNs, the active time is not optimally used and energy will be wasted on idle listening. This observation is the motivation for another milestone in energy-efficient MAC protocols for WSNs, which is described in Section 8.2.3.

8.2.3 Timeout-MAC

The Timeout-MAC (T-MAC) protocol [Dam03] reduces idle listening by transmitting all messages in bursts of variable length and sleeping between bursts. Similarly to what happens with S-MAC, T-MAC nodes form a virtual cluster to synchronize themselves on the beginning of a frame, but instead of using a fixed-length active period, T-MAC dynamically adapts the duty cycle to the network traffic. To determine the end of the active period T-MAC uses a timeout mechanism, in which the timeout value at each node, TA, is set to span a small contention period and an RTS/CTS exchange. If a node r does not detect any activity (i.e., an incoming message or a collision) within an interval TA, it will conclude that no neighbor is going to communicate with it and enter into the sleep mode. Otherwise, the node r will start a new timeout after the on-going communication finishes. A comparison between the different approaches adopted by S-MAC and T-MAC is shown in Figure 8.3.

In [Hal05] T-MAC protocol has proven to adapt seamlessly to traffic fluctuations typical of sensor network applications at the expense of a decreased maximum throughput. T-MAC performs slightly better for variations over time (events) than for variations in location (periodic reporting). Implementation of the T-MAC protocol revealed that, under high workloads, nodes communicate without sleeping, while under light workloads nodes use their radios for as little as 2.5% of the time, saving much more energy as compared to the S-MAC protocol.

A drawback of T-MAC is the early-sleeping problem, that occurs when a node r does not detect any activity during the TA interval not because there is no node interested in communicating with it, but because such a node, s, lost the contention with a third node n which is not a common neighbor of the two. When this occurs, s remains silent and r goes to sleep. After n's transmission finishes, s will be able to send an RTS to r, but being the latter in the sleep state, s will not receive the matching CTS and must wait until the next frame before retrying. To mitigate this problem, T-MAC provides two mechanisms, the future request-to-send (FRTS) and the full-buffer priority options [Dam03]. The FRTS mechanism provides for sending an FTRS packet when a CTS destined for another node has been overheard, so that its destination is advised to stay awake. The full-buffer priority solution gives priority to a node with full buffer. This can be done by sending a new RTS to another node in response

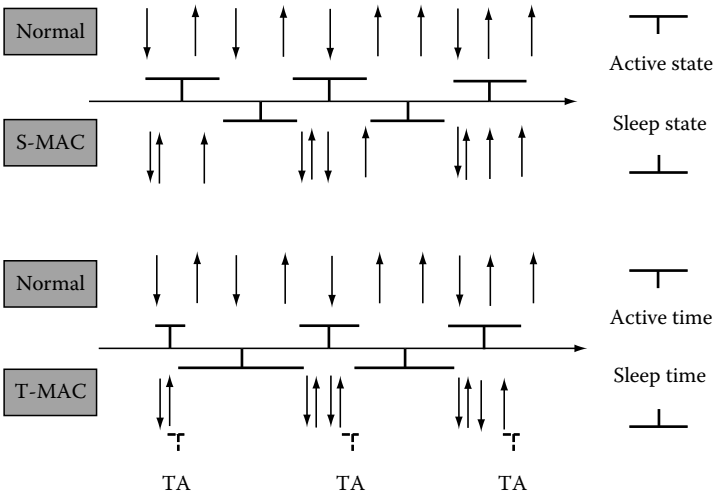


FIGURE 8.3 Duty cycles comparison between S-MAC and T-MAC. The arrows represent message transmissions and receptions.

to the received RTS instead of the usual CTS. This solution may cause performance degradation under high loads, so it has to be used carefully.

8.2.4 Data-Gathering MAC

MAC protocols for multi-hop WSNs that utilize listen/sleep duty cycles suffer from the so-called data forwarding interruption problem, because not all the nodes on a multi-hop path to the sink are aware of the forthcoming arrival of on-going data. As a result, some of them go to sleep, thus introducing a significant latency due to sleep delay.

The Data-Gathering MAC (DMAC) [Lu04] is an energy-efficient low-latency protocol designed and optimized to solve the data forwarding interruption problem in convergecast WSNs. The key idea of DMAC in order to enable a continuous packet flow from sensor nodes to the sink is to stagger the wake-up scheme by giving the sleep schedule of a node an offset, which is a function of its level on the data-gathering tree. During the receive period of a node, all of its child nodes have transmit periods and contend for the channel. To reduce collisions during the transmit period of nodes on the same level in the tree, every node backs-off for a period plus a random time within a fixed contention window before packet transmission. DMAC can be viewed as an extension of the Slotted Aloha algorithm [Rob72] in which slots are assigned to the sets of nodes based on a data-gathering tree.

Low node-to-sink latency is obtained by assigning subsequent slots to the nodes that are consecutive in the data transmission path. Similarly to T-MAC, DMAC also adjusts the duty cycles adaptively according to the network workload.

DMAC fits well the scenarios in which data transmission paths are known in advance (so it is possible to build the data-gathering tree) and outperforms S-MAC in terms of latency (thanks to staggered schedules), throughput and energy efficiency (thanks to the adaptivity). However, when collisions are likely to occur, i.e., in event-triggered WSNs, DMAC performance degrades.

8.2.5 Berkeley MAC

The Berkeley MAC (B-MAC) protocol [Pol04] is a carrier-sense configurable MAC protocol for WSNs that provides a flexible interface to combine low-power operation, effective collision avoidance, and high channel utilization. B-MAC consists of four main components, i.e.,

- Clear channel assessment (CCA) and packet back-off, for channel arbitration
- Link layer ACKs, for reliability
- LPL, for low-power communication

CCA is performed using a weighted moving average on a set of samples taken when the channel is idle to assess the ground noise, thus enabling a better detection of valid transmissions and collisions. The approach has proven to outperform thresholding methods used in a variety of wireless protocols which produce a large number of false negatives that reduce the channel bandwidth.

Packet back-off is configurable and is chosen from a linear range instead of the usual exponential back-off approach. Optional packet-by-packet link layer ACKs are provided to reliably transfer important data. Finally, low-power operation is achieved through an adaptive preamble sampling scheme able to reduce duty cycle and minimize idle listening. A node cycles between awake and sleep states. While awake, it listens a long preamble to assess whether it has to remain awake or can go back to sleep. RTS/CTS packets, due to their high overhead, are not used.

B-MAC supports on-the-fly reconfiguration and provides bidirectional interfaces for services to optimize performance in terms of throughput, latency, or power saving. While S-MAC and T-MAC are not just link protocols, but also network and organization protocols, the lightweight Berkeley MAC protocol only contains a small core of media access functionalities. Network services such as organization, synchronization, and routing build upon it. As B-MAC is very configurable, other MAC protocols for WSNs (e.g., S-MAC and T-MAC) may be implemented efficiently upon B-MAC and could benefit from its flexibility.

B-MAC has a set of interfaces that allow choosing various services to tune its operation in addition to the standard message interfaces. These interfaces allow network services to adjust CCA, ACKs, back-offs, and LPL. Through such interfaces, protocols built on B-MAC can make local policy decisions to optimize power consumption, latency, throughput, fairness, or reliability.

8.2.6 WiseMAC

WiseMAC [Hoi04] is a CSMA-based protocol which uses the preamble sampling technique [Hoi02] to reduce the cost of idle listening. In the preamble sampling technique, nodes regularly sample the channel for a short interval to assess whether there is on-going activity or not. All nodes in the network sample the channel with the same constant period, which is independent of the actual traffic, while their relative sampling schedule offsets are independent. If a node finds the channel busy, it will keep on listening until either a data packet is received or the channel becomes idle again. The transmitter transmits a wake-up preamble in front of every message to ensure that the receiver will be awake when the data portion of the message will arrive. Such a preamble, however, introduces a power consumption overhead, both at the transmitter and at the receiver. In WiseMAC the length of the wake-up preamble is dynamically chosen by a node exploiting the knowledge of the sampling schedule of its direct neighbors, which is received at every data exchange as it is piggybacked in ACK messages. Every node therefore maintains an updated table with the sampling schedule offset of its direct neighbors. Another factor which is taken into account when dynamically sizing the preamble length in WiseMAC is the potential clock drift between the sender and the receiver. In [Hoi03] the calculation for the lower bound is presented.

The varying preamble length makes the WiseMAC protocol adaptive to the traffic load. WiseMAC achieves energy efficiency and also provides for scalability and mobility support, as only local synchronization information is used. A performance comparison in [Enz04] showed that WiseMAC is able to provide both low average power consumption in low traffic conditions and high energy efficiency in high traffic conditions, while T-MAC does not have this property. The main shortcoming of WiseMAC is the decentralized sleep-listen scheduling, which determines different sleep and

wake-up times for each neighbor of a node. This problem is particularly significant in broadcast communications.

8.2.7 Power-Efficient and Delay-Aware MAC for Sensor Networks

The Power-Efficient and Delay-Aware MAC for Sensor Networks (PEDAMACS) [Erg06] is a TDMA-based scheme that extends the single-hop TDMA to a multi-hop sensor network, using a high-powered access point (AP) to synchronize the nodes and to schedule their transmissions and receptions. The main assumption in PEDAMACS is that the AP has enough power to communicate with all the sensor nodes in one hop, while packets from sensor nodes must travel over several hops to reach the AP. Another assumption is that the nodes periodically generate data to be transmitted to the AP. Network topology is discovered by PEDAMACS automatically.

PEDAMACS works according to four phases:

- *Topology learning*: Each node identifies its (local) topology, i.e., its neighbors, interferers, and its parent node in the routing tree rooted at the AP, obtained according to some routing metric.
- *Topology collection*: Each node sends its local topology information to the AP so that, at the end of this phase, the AP knows the full network topology.
- *Scheduling*: The AP broadcasts a schedule which each node follows, going to sleep during the time-slots when it is not supposed to transmit a packet or to listen to one.
- *Adjustment*: This phase is triggered as necessary to learn the local topology information that was not discovered during the topology learning phase or to discover changes.

The PEDAMACS scheduling algorithm runs in polynomial time and guarantees a delay proportional to the number of packets in the WSN to be transferred to the AP in each period. In [Erg06] PEDAMACS is shown to dramatically outperform both random-access schemes without sleep cycles and S-MAC with two different duty cycles. The PEDAMACS framework can be generalized in many ways and extended to the cases of event-driven data generation and existence of multiple APs.

8.2.8 Traffic-Adaptive Medium Access

The traffic-adaptive medium access (TRAMA) [Raj03] [Raj06] provides energy-efficient collision-free channel access in WSNs exploiting transmission schedules to avoid collisions of data packets at the receivers and an adaptive power switching policy to dynamically put nodes into low-power mode whenever they are not transmitting or receiving.

TRAMA assumes a single, time-slotted channel for both data and signaling transmissions. Time is divided into sections of random- and scheduled-access periods. Random-access slots are used for signaling, while scheduled-access slots are used for transmission. During the random-access period, nodes go through contention-based channel access and thus signaling packets may experience collisions. On the contrary, transmission slots are used for collision-free data transmission and also for schedule propagation. Given the low data rates of a typical WSN, the duration of time-slots is much larger than typical clock drifts, so a timestamp mechanism is enough for node synchronization.

TRAMA consists of three components:

- Neighbor Protocol (NP)
- Schedule Exchange Protocol (SEP)
- Adaptive election algorithm (AEA)

The NP and SEP allow nodes to exchange two-hop neighbor information and their schedules. The AEA uses this information to select the transmitters and receivers for the current time-slot, thus enabling noninvolved nodes to switch to a low-power mode.

NP propagates one-hop neighbor information among neighboring nodes during the random-access period using the signaling slots. SEP is used to exchange schedules with neighbors, where a schedule contains information on traffic coming from a node, i.e., the set of receivers for the traffic originating at the node. Before transmitting, a node has to announce its schedule using SEP. This mechanism provides a consistent view of schedules across neighbors and periodic schedule updates.

AEA selects transmitters and receivers to achieve collision-free transmission using the information obtained from NP and SEP. AEA uses traffic information (i.e., which sender has traffic for which receivers) to improve channel exploitation.

TRAMA provides support for unicast, broadcast, and multicast traffic and differs from S-MAC for two main aspects:

- TRAMA is inherently collision-free, as its MAC mechanism is schedule-based, while S-MAC is contention-based.
- TRAMA uses an adaptive approach based on current traffic to switch nodes to low-power mode, while S-MAC is based on a predefined static duty cycle.

In [Raj03] simulation results showed that TRAMA achieves higher end-to-end throughput and energy saving than S-MAC and other contention-based MAC protocols, but also higher delays than random-selection protocols due to the scheduling overhead.

8.2.9 Flow-Aware Medium Access

The Flow-Aware Medium Access (FLAMA) protocol [Raj05] uses a simple traffic adaptive, distributed election scheme for energy-efficient channel access. Similarly to TRAMA, it requires two-hop neighborhood information. In addition, FLAMA uses flow information in the neighborhood to perform the election. FLAMA adapts medium access schedules to the application traffic flows. As the adaptive scheme is simple, it can be executed even by nodes with limited resources and processing capabilities. Similarly to TRAMA, FLAMA assumes a single channel for data and signaling and divides time into contention-based channel access periods, called random-access intervals, and collision-free channel access periods, called scheduled-access intervals. Random-access intervals are used for neighbor discovery, time synchronization, and implicit traffic information exchange, while scheduled-access intervals are used for data transmission. Periodic random-access periods allow FLAMA to adapt to topology and traffic changes in the WSN. Unlike TRAMA, FLAMA does not require explicit schedule announcements during scheduled access periods. Instead, nodes exchange application-specific traffic information during random-access interval to acquire the application-specific traffic flows. FLAMA then exploits the information to calculate transmission schedules for each node. FLAMA achieves adaptivity to the varying traffic conditions by assigning slots to a node depending on the amount of traffic generated by that node, using weights that depend on the amount of the node's incoming and outgoing traffic flows. As nodes with more outgoing flows receive higher weights, they are assigned more slots.

8.2.10 Sift

The Sift protocol proposed in [Jam06] is a MAC protocol for event-driven WSNs that is based on the observation that in many WSN scenarios spatially correlated contention causes latency and throughput degradation. Such a contention occurs when multiple nodes located in the same neighborhood sense the same event and transmit to report it to the sink. In many WSN applications, however, not all the nodes that sense an event need to report it. The first R of N potential reports are the most crucial

ones, which need to be relayed with low latency. To achieve this requirement, the Sift protocol uses a nonuniform probability distribution function of picking a slot within the slotted contention window. If no node begins to transmit in the first slot of the window, then each node increases its transmission probability exponentially for the next slot, assuming that the number of competing nodes is small. Sift obtains a very low latency, but at the expense of increased energy consumption due to idle listening and overhearing, thus trading energy efficiency for latency. Moreover, it requires system-wide time synchronization.

8.2.11 Lightweight Medium Access

The Lightweight Medium Access (LMAC) [Hoe04] protocol aims to minimize the overhead of the physical layer reducing the number of transceiver state switches and the energy waste due to the preamble transmissions. LMAC uses TDMA to provide WSN nodes with a collision-free channel access. Unlike traditional TDMA-based protocols, the time-slots in LMAC protocol are not assigned to nodes by a central authority, but through a fully distributed slot assignment mechanism. Slots are organized in frames. Each node is given one time-slot per frame, during which the node will transmit a message consisting of two parts, a control message and a data unit.

The control message, which has a fixed size, carries:

- ID of the time-slot controller
- Distance (in hops) of the node to the gateway for routing purposes
- Intended receiver(s)
- Length of the data unit

The control message is also used to maintain synchronization between the nodes, as it contains the sequence number of the time-slot in the frame.

All nodes listen to the control messages of their neighboring nodes. When a node is not the recipient of that message and the message is not a broadcast one, the node will switch off its transceiver only to wake at the next time-slot. Otherwise, if the node is the intended receiver, it will listen to the data unit. If the latter does not fill up the remaining portion of the time-slot, after the message transfer has completed, both transmitter and receiver(s) turn off their transceivers.

A short timeout interval ensures that nodes do not waste energy due to idle listening to free time-slots. To limit the number of time-slots in the network, slot reusing is allowed at a noninterfering distance. It is required that a slot is not reused within a two-hop neighborhood, so LMAC includes a bitmap with all the slots assigned to a node's neighbors in the header. By combining the bitmaps of all its neighbors, a node can assess which slots are free within a two-hop neighborhood.

Simulation results in [Hoe04] show that LMAC significantly improves the network lifetime as compared to S-MAC.

8.2.12 Z-MAC

Z-MAC [Rhe05] is a hybrid MAC protocol for WSNs that tries to combine the positive features of TDMA and CSMA scheme. In Z-MAC, nodes are assigned time-slots using a distributed algorithm, but, unlike in typical TDMA mechanisms, a node can transmit in both its own time-slots and slots assigned to other nodes. However, owners of the current time-slot always have priority in accessing the channel over nonowners. The priority is implemented by adjusting the initial back-off period, so that higher priority nodes have shorter back-off periods. The result is that during the slots where owners have data to transmit, Z-MAC reduces their probability of collision, whereas when owners do not have data to transmit, nonowners can steal the slot. This mechanism allows

switching between CSMA and TDMA depending on the current level of contention (CSMA under low contention, TDMA under high contention).

Two different modes of operation are defined, the low contention level (LCL) and the high contention level (HCL).

Under LCL nonowners are allowed to compete in any slot with low priority, while under HCL a node does not compete in a slot owned by its two-hop neighbors.

Z-MAC is robust to topology changes and clock synchronization errors. A local synchronization scheme, where each sending node adjusts the synchronization frequency based on its current data rate and resource budget, is used. In the worst case, i.e., when clocks are completely unsynchronized, Z-MAC performance is comparable to that of CSMA. Under high contention, as more transmissions occur, time synchronization becomes more accurate and Z-MAC performance approaches that of TDMA.

8.2.13 Pattern MAC

The Pattern MAC (PMAC) protocol defined in [Zhe05] is another adaptive MAC protocol for WSNs that tries to enhance the performance of fixed duty cycle protocols, such as S-MAC [Ye02] [Ye04]. Here, the determination of the duty cycle for a node is based on its own traffic and on the traffic patterns of the neighboring nodes. PMAC is a time-slotted protocol but, unlike classical schedule-based MAC protocols for WSNs, it is not based on schedules. Instead, it is based on patterns. A pattern is a binary string that indicates the sleep–wake-up schedule of a node planned for several frames. The basic idea of this protocol is that nodes get information about the activity of their neighbors, so that they can go to sleep also for several frames if there is no activity, whereas in the case that any activity is present, a node knows when it has to wake-up thanks to the patterns. Hence, a node running the PMAC protocol adjusts the sleep–wake-up schedule based on its own pattern and the patterns of its neighbors. In order to achieve high energy saving, the number of sleep times in a pattern is increased exponentially when the network is under light traffic conditions, whereas the sleep sequence is interrupted when a node has data to send. In order to efficiently support the pattern exchange, time is divided into super frame times (SFT), each divided into two parts: pattern repeat time frame (PRTF) and pattern exchange time frame (PETF). Both parts are divided into slots. During the former, nodes transmit data repeating their current patterns. In addition, at the end of the PRTF there is a time-slot during which all nodes stay awake, to speed up communications as well as to support broadcast traffic. On the other hand, the PETF is used for performing pattern exchange between neighboring nodes. To allow every node to send its pattern, the PETF features as many slots as the maximum estimated number of neighbors a node may have.

The PMAC protocol achieves very low energy consumption when the network load is low, as only the sensor nodes involved in the communications will wake-up frequently, whereas the other nodes stay asleep for longer times. This reduces the energy waste due to idle listening as compared to other approaches such as, S-MAC or T-MAC that periodically wake-up nodes. This protocol requires time synchronization, but loose synchronization schemes may be used, e.g., through periodical SYNC packets from neighbors.

8.2.14 Crankshaft

The Crankshaft protocol proposed in [Hal07] is especially designed for WSNs featuring high node density. Under such scenarios, protocols featuring communication grouping, such as S-MAC [Ye02] [Ye04] or T-MAC [Dam03], suffer from degradation in both performance and energy efficiency due to contention and collisions. Moreover, the adaptive duty cycle in T-MAC may prevent nodes to go to sleep. Nevertheless, TDMA-like protocols such as LMAC [Hoe04] would require frames with a large number of time-slots, thus causing a significant delay increase. Moreover, most of the time-slots

may be unused, so the channel might be not fully utilized. Other classical MAC protocols for WSNs, such as PMAC [Zhe05] or WISEMAC [Hoi04], store neighboring information, so in dense WSNs the memory requirements on sensor nodes might be undesirably high, or only a small subsection of neighbor information could be maintained.

The Crankshaft protocol tries to solve the problems that classical schedule-based MAC protocols for WSNs have when they are applied to dense WSNs. Both overhearing and communication grouping are reduced by alternating the power-off of sensor nodes, while the bandwidth exploitation is enhanced by using receive-slots instead of send-slots. In addition, the proposed mechanism does not require per-node neighborhood information. The basic idea of the protocol is that nodes should stay awake listening for incoming packets at fixed offsets from the start of a frame. The name comes from the analogy with engines, where the time when a piston fires is a fixed offset from the start of the crankshaft rotation.

The Crankshaft protocol divides time into frames that are further divided into slots. Two types of slots are defined: unicast and broadcast. Every node has to be awake during the broadcast slots in order to receive broadcast communications. Instead, each node listens for only one unicast slot in every frame. The only exception is the sink node that stays awake during all the time-slots of the frame. A node willing to transmit a packet contends for the medium access, either during a broadcast slot, or during the unicast slot in which the destination node is active. This is because the sink node is assumed to be less energy-constrained than the other nodes, and also because most of the traffic in a WSN is typically destined to it. Nodes do not need explicit slot assignation, as a very simple criterion is used to decide the listening time-slot. If n is the number of the time-slots inside a frame, the listening time-slot of each node is obtained by calculating MAC address modulo n. In order to improve energy efficiency even further, each time-slot is divided into two different parts, i.e., the contention window and the message exchange window. The source nodes have to wake-up during the former and resolve the contention, while the destination wakes-up in the second part. This protocol can achieve good energy efficiency and good convergecast delivery ratio in dense WSNs, but with high latency values. In addition, the rigid structure of the protocol may limit its applicability.

8.2.15 Correlation-Based Collaborative MAC

A different approach that achieves energy efficiency exploiting spatial data correlation is presented in [Vur06]. This protocol, called a spatial Correlation-based Collaborative MAC (CC-MAC), is designed for event-driven WSNs. Usually, in such networks, data from spatially separated sensor nodes is more useful than highly correlated data from close nodes. Reducing the transmission attempts of correlated (and redundant) data it is possible to save energy, bandwidth, and time. So, the basic idea of the CC-MAC protocol is to intelligently manage the transmissions, taking into account the spatial correlated nature of the event information. In fact, it is not necessary that all the nodes sensing an event transmit their data to the sink, but a smaller set of measurements may be sufficient. However, as the number of the nodes that transmit their data is reduced, the information decoded by the sink is degraded. So, also the reliability of the event detection is reduced. The aim of the CC-MAC protocol is to exploit the spatial correlation to reduce the transmission attempts without compromising the event detection performance, by introducing a distortion constraint to be met. Thus, the minimum number of representative nodes that achieve the distortion constraint has to be selected. Intuitively, the minimum distortion is achieved when the representative nodes are located close to the event source, but as far from each other as possible. Another benefit of selecting representative nodes far from each other is that spatial wireless channel reuse can be achieved. In order to exploit the spatial correlation in a distributed fashion, an Iterative Node Selection scheme is proposed, that exploits the statistical properties of the node distribution to compute the correlation radius, r_{corr}. This algorithm is executed during the network initialization phase. The computed radius is then used by the distributed MAC mechanisms. Two different MAC mechanisms are defined for event detection (E-MAC) and packet forwarding (N-MAC), respectively. They are both based on the

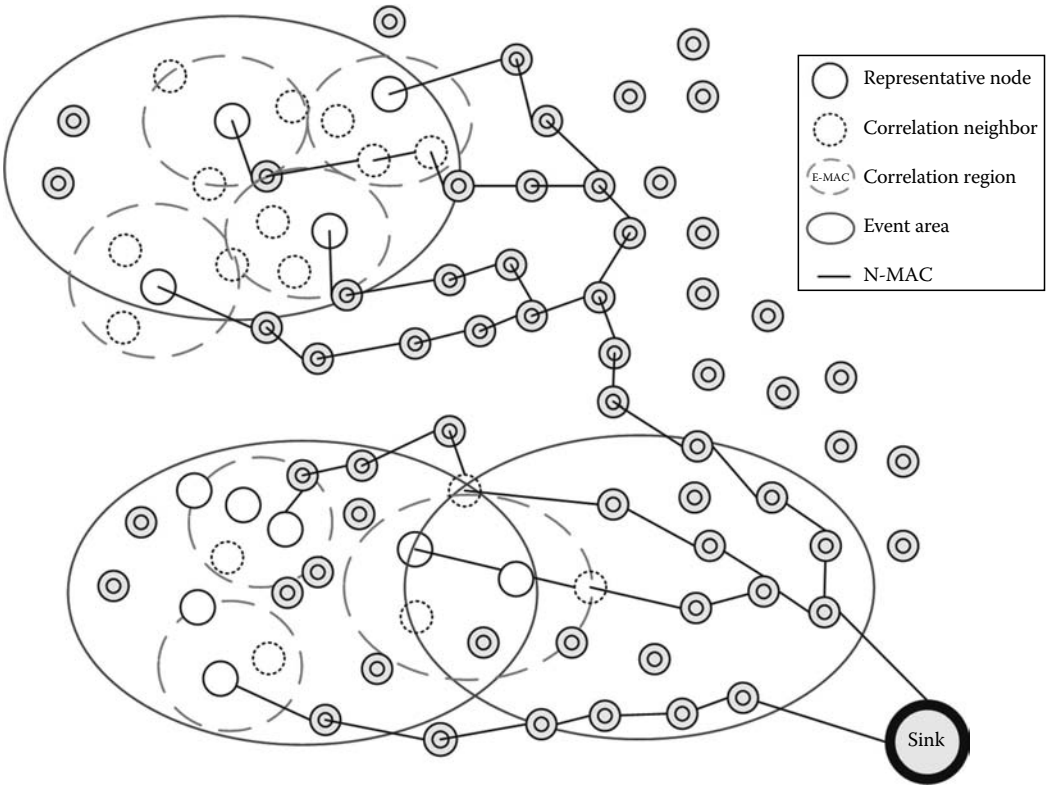


FIGURE 8.4 CC-MAC protocol operations. (Redrawn from Vuran, M.C. and Akyildiz, I.F., *IEEE/ACM Trans. Network.*, 14(2), 316, 2006.)

CSMA/CA protocol and use RTS/CTS packets to regulate the channel access. The aim of E-MAC is to filter out correlated data, while the aim of N-MAC is to give the forwarding packets priority over the event detection packets. In fact, in order to meet the desired distortion, all the packets filtered by E-MAC should be received by the sink, so they have to be prioritized. With the E-MAC mechanism one node at a time can transmit, the representative node, whereas the others stop the transmission attempts for the duration of the transmission. After the transmission occurs, a new representative node may be selected. The selection is performed through a simple RTS/CTS/DATA/ACK sequence. A representation of the E-MAC and N-MAC operations is given in Figure 8.4.

The nodes hearing the data transmission determine if they belong to the same correlation region, i.e., if their distance from the representative node is smaller than r_{corr}. If this is the case, they stop attempting to transmit; otherwise they enter the contention for the medium when a packet has to be sent. When a representative node is selected, the other nodes belonging to the correlation region go to sleep. However, they periodically wake-up in order to check if they have to perform some data forwarding. Furthermore, in the case r_{corr} is larger than the transmission range, the nodes belonging to the same correlation region can go to sleep only if they do not belong to the path toward the sink node, otherwise the event detection capabilities may be seriously affected. The N-MAC mechanism is the classical CSMA/CA, but it is given priority over the E-MAC through the use of smaller values for interframe spacing and CW_{max}.

This protocol has been evaluated through simulations and the results show that it performs better than S-MAC, T-MAC, and TRAMA in terms of latency and energy consumption, thanks to the reduced number of packets together with the decreased nodes' duty cycle. The packet drop ratio on

the useful data is reduced, as the packets are early filtered. However, the experienced distortion is generally worse than the desired value, because, although the packet drop ratio is reduced, packet drop may still occur. As it is also not predictable, the only way to deal with the problem is by assessing the performance of the WSN through simulations and correcting the distortion requirements. Finally, as this approach addresses only collective requirements, it is not possible to meet performance on a per-packet basis. So, CC-MAC is not suitable for networks in which QoS requirements should be met on each packet.

8.3 Mobility Support in WSNs

In WSNs two kinds of mobility can be defined, i.e., weak and strong mobility. Weak mobility refers to network topology changes following either node failures (due to hardware failures or battery consumption) or node joins (due to the addition of new sensor nodes to cover a larger sensing area or to improve the network lifetime). Weak mobility can be found even in sensor networks with static nodes. Strong mobility refers to physical mobility combined with concurrent node joins/failures. WSN nodes may change their physical location for several reasons. For example, they may be moved by external weather conditions (such as, wind, rain, or waves) or following some mechanism introduced with the aim of enhancing the network performance [Kan04] or lifetime [Wan05]. Another typical case is sink mobility, which occurs when the sink nodes are handheld devices used to perform data queries and to collect results, as in [Luo05]. In some cases it is the sensing application that requires mobile sensor nodes. Examples can be found in WSNs for health care or military applications based on wearable sensors.

Classical MAC protocols for WSNs usually use schedule-based approaches in order to decrease the nodes' duty cycles and thus energy consumption. Such approaches are very effective in stationary networks, where connections are setup and closed not so frequently, but they may be not adequate in scenarios featuring high mobility, as the overhead of scheduling calculation may lead to poor network performance. In addition, with node mobility the two-hop neighborhood information collected by nodes may become inconsistent for a long period, thus affecting the correctness of the protocol. On the other hand, contention-based MAC protocols are more reactive to topology changes, as no schedules have to be recalculated. However, such protocols are generally less energy-efficient than schedule-based ones, as it is harder to decrease the nodes' duty cycles without affecting the correctness of the protocol operations. For this reason some MAC protocols specifically designed for WSNs with mobility support have been proposed. Such protocols try to find a trade-off between energy efficiency and performance under mobility conditions, using adaptive or hybrid approaches. In the following, some notable protocols are briefly described.

8.3.1 Mobility-Aware MAC for Sensor Networks

The mobility-aware MAC protocol for WSNs described in [Pha04] aims to provide a medium access mechanism that works efficiently on WSNs both with and without node mobility (as shown in Figure 8.5). The Mobility-Aware MAC for Sensor Networks (MS-MAC) protocol extends the well-known SMAC protocol to support mobile sensor nodes through an adaptive mobility handling mechanism which changes the duty cycle of nodes as a function of their estimated mobility.

This protocol works in an energy-efficient SMAC-like fashion when nodes are stationary, while it works similarly to IEEE 802.11 in the presence of mobility. Nodes estimate the presence of mobility within their neighborhood through the received signal strength of the periodical SYNC frames. When a variation on the signal level of a neighbor is detected, the node is assumed to be moving. When a mobile node is detected, the mobility information is included in the periodical SYNC frames.

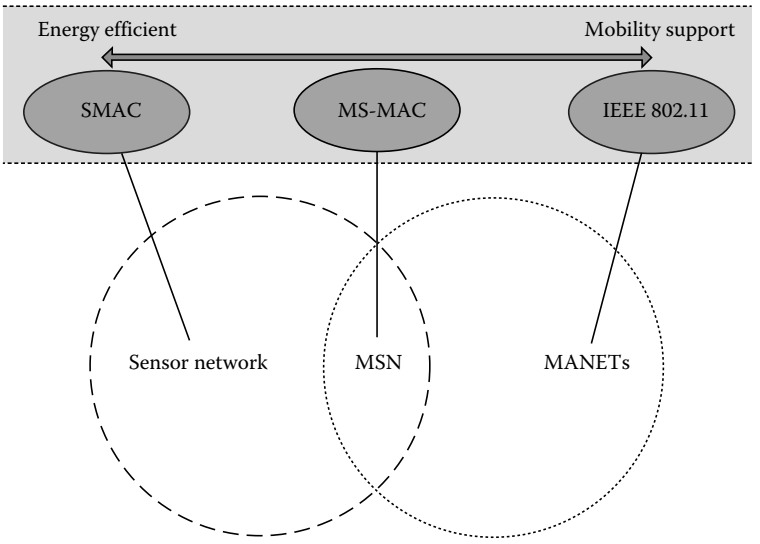


FIGURE 8.5 MS-MAC and mobility in WSNs. (Redrawn from Pham, H. and Jha, S., An adaptive mobility-aware MAC protocol for sensor networks (MS-MAC), in *Proceedings of the 1st IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS-2004)*, Fort Lauderdale, FL, Oct 2004.)

However, when multiple mobile nodes are detected, only information on the node with the highest estimated speed is included. The mobility information is used to create a so-called active zone around the moving node that is a region in which the synchronization period is smaller. This means that nodes within the active zone have a higher energy consumption, but also that the time needed to setup a new connection is shorter. This mechanism highly improves the performance of the WSN under mobility. In fact, using the standard synchronization periods, a mobile node might lose the connection with the old neighbors before it has setup a new schedule. The node might stay therefore disconnected for a long time, until a new synchronization phase starts (the default period is 2 min). On the other hand, decreasing the synchronization periods, a node can speed up connection setup, so that a new schedule can be created before the old connections are lost. Under stationary conditions no active regions are created, so the node features low energy consumption, whereas when the mobility of a node is detected, only the surrounding nodes increase their synchronization rates. The nodes in the active zone adjust their duty cycles depending on the estimated speed of the moving nodes and will also stay awake all the time if a threshold speed is exceeded. This way, a trade-off between performance and energy consumption is obtained in both stationary and mobile scenarios.

8.3.2 Mobility-Adaptive Collision-Free MAC

A different approach to handle mobility is used by the Mobility-Adaptive Collision-Free MAC (MMAC) protocol proposed in [Mun05]. This protocol uses a dynamic frame containing both scheduled-access and random-access time-slots. The division between scheduled-access and random-access slots is adaptively changed according to the expected mobility changes, e.g., node joins or node failures. The MMAC protocol assumes that WSN nodes are location-aware. The location information is used to predict the mobility patterns of nodes, through a real-time mobility estimation scheme (AR-1) based on a first-order autoregressive model [Zai04]. The key idea is to use the information obtained from the mobility estimation model to reduce the frame time when a large number of nodes is expected to enter or leave the two-hop neighborhood, and vice-versa to increase the frame time when less node mobility is expected. The main issues of such an approach are how to obtain

mobility information about all the current and potential two-hop neighbors, and how to perform synchronization between nodes, as they could independently calculate frame times different from each other. To overcome these issues, this protocol uses a cluster-based approach, in which nodes are grouped into clusters. A cluster-head node for each cluster is elected in rotation, with an election schema similar to the one used in low energy-adaptive cluster hierarchy (LEACH) [Hei02]. Time is divided into rounds, and a different cluster-head is elected at each round. The predicted mobility information is put by nodes into the header of MAC packets. The cluster-head is always on, so it collects all the information and it broadcasts all the mobility information to the member nodes during a dedicated time-slots at the end of the frame. This provides the nodes with a best-effort knowledge of the mobility information of the current and potential two-hop neighborhood. A similar solution is used to provide synchronization between nodes. They calculate independently their frame time, but, instead of setting-up the frame by themselves, nodes communicate their frame time to the cluster-head, which collects the value for all its member nodes and calculates the average value. At the end of each round, cluster-head nodes exchange all the average values between themselves and the global mean value is disseminated along the whole network. So, at the end of each round, all the nodes adjust the frame time as well as the scheduled access slots and the random-access slots according to the global frame time. While the frame time remains the same for the whole round, at the end of each frame the number of random-access slots within a cluster may be increased or decreased, according to the mobility patterns of cluster nodes.

8.3.3 Mobility-Adaptive Hybrid MAC

Another approach to handle mobility in WSNs is the Mobility-Adaptive Hybrid MAC (MH-MAC) presented in [Raj08] that tries to combine the advantages of schedule-based and contention-based protocols. The proposed solution is a hybrid approach in which the WSN nodes are differentiated in mobile nodes and static nodes, and the most suitable medium access mechanism is used for each type of nodes. So, for static nodes a schedule-based channel access mechanism is adopted, while a contention-based approach is used for mobile nodes. Time is divided into time-slots and two different types of time-slots, i.e., static and mobile, are defined. Each node uses a mobility estimation algorithm to determine its mobility. Based on its mobility, a node uses the static or mobile slot. Static slots are assigned in a LMAC-like fashion, while for the contention-based time-slots a Scheduled Channel Polling mechanism [Ye06] is used to limit the duty cycle in order to decrease energy consumption. According to such a mechanism, a sender node sends a short wake-up tone followed by the receiver ID, so that all the other nodes but the receiver can go to sleep. The MH-MAC protocol adapts to different levels of mobility by dynamically adjusting the ratio static/mobile slots as well as the frame time. When less nodes are mobile, more slots are reserved for schedule-based allocation and vice-versa.

8.4 Multichannel Protocols for WSNs

Multichannel protocols are usually hybrid approaches that combine FDMA with TDMA and/or CSMA and use different frequencies for parallel communications. Typically, these approaches try to maintain all the benefits, in terms of energy consumption, of TDMA-based protocols by avoiding contentions and lowering the nodes' duty cycles. In addition, thanks to the use of multiple channels, these protocols enhance the network capacity and solve the scalability problems that affect schedule-based protocols when applied to large and dense WSNs. However, providing multichannel support introduces some overhead. The first overhead is represented by the signaling required to enable communicating nodes to agree on the radio channel to use. The second source of overhead is channel switches, which cause energy consumption and therefore should be limited.

As compared to standard MAC protocols, multichannel protocols also raise new problems [Mah06]. As an example, the multichannel hidden terminal node is harder to solve than the classical hidden terminal node, as control packets (e.g., RTS/CTS) could be sent on different channels, while a network interface is able to work only on one channel at a time. Another problem that causes energy and bandwidth waste on multichannel communications is deafness, which occurs when a node A transmits to node B while the latter is transmitting to node C on a different radio channel. Finally, even the co-existence of unicast and broadcast transmissions can be problematic in a multichannel environment [Zho06b], as it needs either different priorities for broadcast and unicast traffic or the alternation of carrier sensing on the broadcast and unicast channel, to be stopped when a signal is overheard. This technique is called toggle snooping.

The channel assignment may be computed either following a distributed approach or by the sink node in a centralized way and then forwarded to nodes. The majority of protocols use distributed approaches. The assignment may be fixed, as in [Zho06a], or dynamic, as in [Che06]. In the following, some multichannel MAC protocols are briefly discussed.

8.4.1 Multifrequency Media Access Control for WSNs

The Multifrequency MAC for Wireless Sensor Networks (MMSN) protocol consists of two different parts, i.e., a frequency assignment schema and a medium access protocol. The frequency assignment schema aims to assign different frequencies to each neighbor or, if the number of frequencies is not large, to efficiently assign the available frequencies in order to limit the potential communication conflicts. In [Zho06a] four different assignment strategies are proposed, i.e.,

- *Exclusive frequency assignment*: The nodes exchange their IDs with the two-hop neighbors and then make the frequency decision in the increasing order of ID values, so that nodes with lower IDs have the lower frequencies.
- *Even selection*: The exclusive frequency assignment is extended to the case in which all frequencies have been already used by at least one node within two hops. In this case a random frequency is selected among the least used ones.
- *Eavesdropping*: Each node broadcasts its frequency decision and waits for a back-off interval. During such an interval the other decisions are recorded. When the back-off timer expires, a frequency is randomly selected from the least chosen ones. This mechanism only uses one-hop neighbor information, so it features less overhead, but also more conflicts.
- *Implicit-consensus*: The two-hop neighbors' IDs are collected as in exclusive frequency assignment, then each node calculates its frequency locally, using a pseudorandom number generator seeded by node IDs and a defined algorithm. This schema provides different channels for all two-hop neighbors, but it is effective only when the number of available frequencies is large.

One of these schemes can be used to assign the frequency for data reception. Nodes are synchronized and the medium access divides the time into time-slots. A time-slot consists of a broadcast contention period T_{bc}, in which nodes contend for the same broadcast frequency, and a transmission period T_{tran}, in which nodes contend for shared unicast frequencies. The problem of supporting unicast and broadcast transmissions at the same time is solved by using a dedicated radio channel and prioritizing broadcast traffic. Nodes first check the broadcast frequency for receiving or transmitting broadcast packets. If there are no broadcast packets, nodes can transmit or receive a unicast packet. A time-slot can be used either to transmit or to receive one packet. Nodes that have no data to transmit simply listen for broadcast packets, then switch to their receiving frequency and listen for unicast packets. Nodes that have packets to send use a CSMA approach with random back-off to access the medium

during the broadcast or unicast period, according to their requirements. However, in order to detect unicast packets to be received when there are also packets to be sent, toggle snooping and toggle transmission are used, i.e., both listening and transmission are performed alternately on two different frequencies with different rates, so that the detection of a packet to be received is guaranteed within a maximum delay. Simulation results show that MMSN features both reduced energy consumption and higher performance as compared to the standard CSMA mechanism when the number of available channels increases.

8.4.2 Multichannel LMAC

In [Dur06] the problem of providing multiple channel support to an example single-channel MAC protocol, the LMAC protocol [Hoe04], is addressed. The authors propose to multiplex the time-slot in the frequency domain in an on-demand fashion, in order to enhance the spectrum utilization. This approach does not require multiple transceivers, and the switching is performed only when the network reaches its density limit. The proposed technique works in two phases. In the first phase, nodes select their time-slot according to the classic LMAC protocol, while in the second phase nodes select the radio channels. In fact, while in LMAC a time-slot can be reused only after at least two hops, in the multichannel approach proposed in [Dur06] the same time-slot may be reused on a different radio channel. Thus, a node which finds its time-slot already occupied by a two-hop neighbor can use the same time-slot but on a different channel. Bridge nodes are used to maintain the whole network connected. Simulation results shown that applying a similar approach to a classical MAC protocol, such as LMAC, denser networks can be supported. In addition, the number of collisions is significantly decreased.

8.4.3 Multichannel MAC

In [Che06] another Multichannel MAC (MCMAC) protocol for WSNs is presented. Unlike MMSN, this protocol does not use a fixed channel assignment for nodes, but they are dynamically selected. This protocol introduces network clusterization and exploits the existence of cluster-head nodes that collect request messages from the cluster members, select the radio channels and communicate them to both source and destination nodes. In this way, node pairs can communicate using the received schedule and the designated radio channel. Although this approach can increase the sleeping times of nodes, thus lowering their power consumption, it introduces a significant overhead, due to the high number of signaling messages sent from/to the cluster-head.

8.5 Summary and Open Issues

This chapter addressed MAC layer protocols for WSNs from a broad perspective, ranging from static, single-channel MAC protocols to multichannel approaches and protocols able to support mobility. While the MAC protocols are usually grouped into two main categories, schedule-based and contention-based, a number of MAC protocols for WSNs actually try to combine the advantages of both classes in order to cope with the conflicting requirements of WSNs, such as energy efficiency, scalability, timeliness, adaptability, and fault tolerance. Several MAC protocols in the literature are overviewed in this chapter, to give an idea of the problems addressed, of the techniques used and the performance obtained by these solutions. Classical approaches as well as novel trends, such as the use of spatial correlation to enhance energy efficiency, are discussed.

Despite the large number of existing MAC protocols, there is still room for further improvements, in order to achieve better trade-offs between timeliness and energy efficiency. For example, the

multichannel approach is an interesting solution to increase the available bandwidth and decrease contention, so further research might investigate the effect of the interferences as well as the obtainable network capacity. MAC protocols might exploit the benefits of multichannel approaches to obtain bounded delay and good scalability at the same time. Another promising research trend for MAC protocols might be the combination of spatial-correlation awareness with classical energy-efficient techniques to improve the network lifetime even further.

In WSNs, the Application, Transport, Routing, MAC, and Physical layers have common requirements and are highly dependent on each other. As a result, cross-layering, where layers are integrated with each other, is advisable. In addition, as layering of protocols introduces overheads for each layer, even in terms of energy consumption, the integration of the protocol layers is an issue which deserves further investigation.

An interesting research thread is represented by MAC protocols for the so-called WSANs. WSANs may be considered a variation of WSNs [Aky04], in which the devices deployed in the environment act not only as sensors able to sense environmental data, but also as actors* able to react and to affect the environment.

WSANs have some notable differences from WSNs. First, actors are usually resource-rich devices equipped with better processing capabilities, stronger transmission power, and a longer battery life than typical sensor nodes. Second, in WSANs, depending on the application, real-time computation and communication are very important issues, since timely actions have to be performed in the environment after sensing occurs. Third, the number of sensor nodes in WSANs may be in the order of hundreds or thousands, but the number of actors is much lower than the number of sensors. Finally, in order to provide effective sensing and acting, a distributed local coordination mechanism is necessary among sensors and actors. These differences make the existing MAC protocols developed for ad-hoc networks or WSNs not suitable for WSANs.

Another promising field of research is physical computing systems. These systems derive from the confluence of embedded and real-time systems with wireless, sensor, and networking technologies through the seamless integration of computing and physical world via sensors and actuators [Sta05]. This confluence is leading to the deployment of networks for the collaborative processing of physical information. Physical computing systems are typically very large and dense and require real-time operation, dependability, security, safety, efficiency, and adaptivity. As current WSNs protocols do not provide all these features, further research is needed to address the challenges raised by this kind of systems.

References

[Aky04] I. F. Akyildiz and I. H. Kasimoglu, Wireless sensor and actor networks: Research challenges, *Ad Hoc Networks*, 2 (2004), 351–367.

[Che06] X. Chen, P. Han, Q. He, S. Tu, and Z. Chen, A multi-channel MAC protocol for wireless sensor networks, in *Proc. of the 6th IEEE International Conference on Computer and Information Technology, CIT 2006*, Washington, DC, p. 224, Sept 2006.

[Chi06] O. Chipara, Z. He, Q. Chen, G. Xing, X. Wang, C. Lu, J. Stankovic, and T. Abdelzaher, Real-time power-aware routing in sensor networks, in *Proc. of IWQoS 2006, the 14th IEEE International Workshop on Quality of Service*, pp. 83–92, New Haven, CT, Jun 2006.

* The concept of actor is different from the concept of actuator. An actuator is a device that converts an electrical control signal to a physical action. An actor not only is able to act on the environment by means of one or several actuators, but is also a network entity that receives, transmits, processes, and relays data.

[Dam03] T. V. Dam and K. Langendoen, An adaptive energy-efficient MAC protocol for wireless sensor networks, in *Proc. of the 1st ACM Conference Embedded Networked Sensor Systems*, Los Angeles, CA, pp. 171–180, Nov 2003.

[Dem06] I. Demirkol, C. Ersoy, and F. Alagöz, MAC protocols for wireless sensor networks: A survey, *IEEE Communications Magazine*, 44(4) (Apr 2006), 115–121.

[Dur06] O. Durmaz Incel, S. Dulman, and P. Jansen, Multi-channel support for dense wireless sensor networking, in *Proc. of the 1st European Conference on Smart Sensing and Context, EuroSSC 2006*, P. Havinga et al. (Eds.), Enschede, the Netherlands, pp. 1–14. Lecture Notes in Computer Science 4272, Springer-Verlag, Berlin Heidelberg, Oct 2006.

[Enz04] C. C. Enz et al., WiseNET: An ultralow-power wireless sensor network solution, *IEEE Computer*, 37(8) (Aug 2004), 62–70.

[Erg06] S.C. Ergen and P. Varaiya, PEDAMACS: Power efficient and delay aware medium access protocol for sensor networks, *IEEE Transactions on Mobile Computing*, 5(7) (Jul 2006), 920–930.

[Hal05] G.P. Halkes, T. van Dam, and K.G. Langendoen, Comparing energy-saving MAC protocols for wireless sensor networks, *Mobile Networks and Applications*, 10 (2005), 783–791, Springer.

[Hal07] G.P. Halkes and K.G. Langendoen, Crankshaft: An energy-efficient MAC-protocol for dense wireless sensor networks, in *EWSN 2007, European Conference on Wireless Sensor Networks*, pp. 228–244, Delft, the Netherlands, Jan 2007.

[Hei02] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, An application-specific protocol architecture for wireless microsensor networks, *IEEE Transactions on Wireless Communications*, 1(4) (Oct 2002), 660–670.

[Hil02] J. Hill and D. Culler, Mica: A wireless platform for deeply embedded networks, *IEEE Micro*, 22(6) (2002), 12–24.

[Hoe04] L. van Hoesel and P. Havinga, A lightweight medium access protocol (LMAC) for wireless sensor networks, in *Proc. of the 1st International Workshop on Networked Sensing Systems (INSS 2004)*, Tokyo, Japan (2004).

[Hoi02] A. El-Hoiydi, Spatial TDMA and CSMA with preamble sampling for low power ad hoc wireless sensor networks, in *Proc. of ISCC 2002, the 7th International Symposium on Computers and Communications*, pp. 685–692, Taormina, Italy, Jul 2002.

[Hoi03] A. El-Hoiydi, J. D. Decotignie, C. Enz, and E. Le Roux, WiseMAC: An ultra low power MAC protocol for the WiseNET wireless sensor network, in *Proc. of the 1st International Conference on Embedded Networked Sensor Systems (SenSys 2003)*, pp. 302–303, Los Angeles, CA, Nov 2003, ACM Press.

[Hoi04] A. El-Hoiydi and J. D. Decotignie, WiseMAC: An ultra low power MAC protocol for the downlink of infrastructure wireless sensor networks, in *Proc. of the 9th International Symposium on Computers and Communications 2004 (ISCC04)*, vol. 2, pp. 244–251, Washington, DC, Jun 2004.

[Jam06] K. Jamieson, H. Balakrishnan, and Y.C. Tay, Sift: A MAC protocol for event-driven wireless sensor networks, in *Proc. of the 3rd European Workshop on Wireless Sensor Networks (EWSN)*, pp. 260–275, Zurich, Switzerland, 2006.

[Kan04] A. Kansal, M. Rahimi, W. J. Kaiser, M. B. Srivastava, G. J. Pottie, and D. Estrin, Controlled mobility for sustainable wireless sensor networks, in *Proc. of the 1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 04)*, Santa Clara, CA, Oct 2004.

[Kul04] S. S. Kulkarni, TDMA services for sensor networks, in *Proc. of the 24th International Conference on Distributed Computing Systems Workshops (ICDCSW'04)*, pp. 604–609, Tokyo, Japan, Mar 2004.

[Lu04] G. Lu, B. Krishnamachari, and C. Raghavendra, An adaptive energy-efficient and low-latency MAC for data gathering in sensor networks, in *Proc. of the International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*, Santa Fe, NM, Apr 2004.

[Luo05] H. Luo, F. Ye, J. Cheng, S. Lu, and L. Zhang, TTDD: Two-tier data dissemination in large-scale wireless sensor networks, *Wireless Networks*, 11(1–2), (Jan 2005), 161–175.

[Mah06] R. Maheshwari, H. Gupta, and S. R. Das, Multichannel MAC protocols for wireless networks, in *Proc. of the 3rd IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks 2006*, Reston, VA, Sept 2006.

[Mun05] M. Ali, T. Suleman, and Z.A. Uzmi, MMAC: A mobility-adaptive, collision-free MAC protocol for wireless sensor networks, in *Proc. of the 24th IEEE International Performance, Computing, and Communications Conference, IPCCC 2005*, pp. 401–407, Phoenix, AZ, Apr 2005.

[Pha04] H. Pham and S. Jha, An adaptive mobility-aware MAC protocol for sensor networks (MS-MAC), in *Proc. of the 1st IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS-2004)*, Fort Lauderdale, Florida, Oct 2004.

[Pol04] J. Polastre, J. Hill, and D. Culler, Versatile low power media access for wireless sensor networks, in *Proc. of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, Baltimore, Maryland, Nov 2004.

[Raj03] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, Energy-efficient, collision-free medium access control for wireless sensor networks, in *Proc. of ACM SenSys '03*, Los Angeles, CA, pp. 181–192, Nov 2003.

[Raj05] V. Rajendran, J.J. Garcia-Luna-Aceves, and K. Obraczka, Energy-efficient, application–aware medium access for sensor networks, in *Proc. of IEEE MASS 2005, the 2nd IEEE International Conference on Mobile Ad-Hoc and Sensor Systems*, Washington, DC, Nov 2005.

[Raj06] V. Rajendran, K. Obraczka, and J.J. Garcia-Luna-Aceves, Energy-efficient, collision-free medium access control for wireless sensor networks, *Wireless Networks*, 12, (2006), 63–78, Springer.

[Raj08] A. Raja and X. Su, A mobility adaptive hybrid protocol for wireless sensor networks, in *Proc. of 5th IEEE Consumer Communications and Networking Conference, CCNC 2008*, pp. 692–696, Las Vegas, NV, 10–12 Jan, 2008.

[Rhe05] I. Rhee, A. Warrier, M. Aia, and J. Min, Z-MAC: A hybrid MAC for wireless sensor networks, in *Proc. of the 3rd international ACM Conference on Embedded Networked Sensor Systems (SenSys '05)*, pp. 90–101, California, Nov 2005.

[Rob72] L.G. Roberts, ALOHA packet system with and without slots and capture, *ACM SIGCOMM Computer Communication Review*, 5(2) (Apr 1975), 28–42.

[Sin98] S. Singh and C.S. Raghavendra, PAMAS: Power aware multi-access protocol with signalling for ad hoc networks, *Computer Communication Review*, 28(3) (1998), 5–26.

[Sta05] J. A. Stankovic, I. Lee, A. Mok, and R. Rajkumar, Opportunities and obligations for physical computing systems, *IEEE Computer*, 38(11) (Nov 2005), 23–31.

[Vur06] M.C. Vuran and I.F. Akyildiz, Spatial correlation-based collaborative medium access control in wireless sensor networks, *IEEE/ACM Transaction on Networking*, 14(2) (Apr 2006), 316–319.

[Wan05] Z.M Wang, S. Basagni, E. Melachrinoudis, and C. Petrioli, Exploiting sink mobility for maximizing sensor networks lifetime, in *Proc. of the 38th Annual Hawaii International Conference on System Sciences, HICSS '05*, p. 287.1, Big Island, HI, Jan 2005.

[Ye02] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks, in *Proc. of the 21st Conference of the IEEE Computer and Communications Societies (INFOCOM 02)*, vol. 3, pp. 1567–1576, New York, Jun 2002.

[Ye04] W. Ye, J. Heidemann, and D. Estrin, Medium access control with coordinated adaptive sleeping for wireless sensor networks, *IEEE/ACM Transactions on Networking*, 12(3) (Jun 2004), 493–506.

[Ye06] W. Ye, F. Silva, and J. Heidemann, Ultra-low duty cycle MAC with scheduled channel polling, in *Proc. of the 4th International Conf. on Embedded Networked Sensor Systems (Sensys)*, Boulder, Colorado, 2006.

[Zai04] Z. R. Zaidi and B. L. Mark, Mobility estimation for wireless networks based on an autoregressive model, in *Proc. of the IEEE Global Telecommunications Conference Globecom 2004*, Vol. 6, Dallas, Texas, pp. 3405–3409, Dec 2004.

[Zhe05] T. Zheng, S. Radhakrishnan, and V. Sarangan, PMAC: An adaptive energy-efficient MAC protocol for wireless sensor networks, in *Proc. of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, pp. 65–72, Denver, CO, 2005.

[Zho06a] G. Zhou, C. Huang, T. Yan, T. He, J.A. Stankovic, and T.F. Abdelzaher, MMSN: Multi-frequency media access control for wireless sensor networks, in *Proc. of INFOCOM 2006, the 25th IEEE International Conference on Computer Communications*, Barcelona, Spain, pp. 1–13, Apr 2006.

[Zho06b] G. Zhou, J. Stankovic, and S. Son, The crowded spectrum in wireless sensor networks, in *Proc. of the 3rd Workshop on Embedded Networked Sensors (EmNets 2006)*, Cambridge, MA, May 2006.

9

Distributed Signal Processing in Sensor Networks

9.1 Introduction 9-1
Notation
9.2 Case Study: Spectrum Analysis Using Sensor Networks 9-3
Background • Estimating the Power Spectrum of a Signal Source Using Sensor Network Data
9.3 Inverse and Ill-Posed Problems 9-6
Ill-Posed Linear Operator Equations • Regularization Methods for Solving Ill-Posed Linear Operator Equations
9.4 Spectrum Estimation Using Generalized Projections 9-10
9.5 Distributed Algorithms for Calculating Generalized Projection 9-12
Ring Algorithm • Star Algorithm
9.6 Conclusion 9-18
References 9-19

Omid S. Jahromi
Sonavation Inc.

Parham Aarabi
University of Toronto

9.1 Introduction

Imagine a networked sensing system with thousands or millions of independent components, all capable of generating and communicating the data. A sensing system of such complexity would seem unthinkable a few decades ago but, today, it has become a possibility, thanks to the widespread availability of cheap embedded processors and easily accessible wireless networks. Networking a large number of autonomous sensing devices is an emerging technology that promises an unprecedented ability to monitor the physical word via a spatially distributed network of small, inexpensive, wireless devices that have the ability to self-organize in a well-connected network. A wide range of applications of sensor networks are being envisioned in a number of areas, including geographical monitoring, inventory management, homeland security, and health care.

The building blocks of a sensor network, often called "Motes," are self-contained, battery-powered computers that measure light, sound, temperature, humidity, and other environmental variables (Figure 9.1). Motes are inevitably constrained in processing speed, storage capacity, and communication bandwidth. Additionally, their lifetime is determined by their ability to conserve power.

In principle, a distributed network of sensors can be highly scalable, cost effective, and robust with respect to individual Mote's failure. However, there are many technological hurdles that must be overcome for sensor networks to become viable. Motes are inevitably constrained in processing

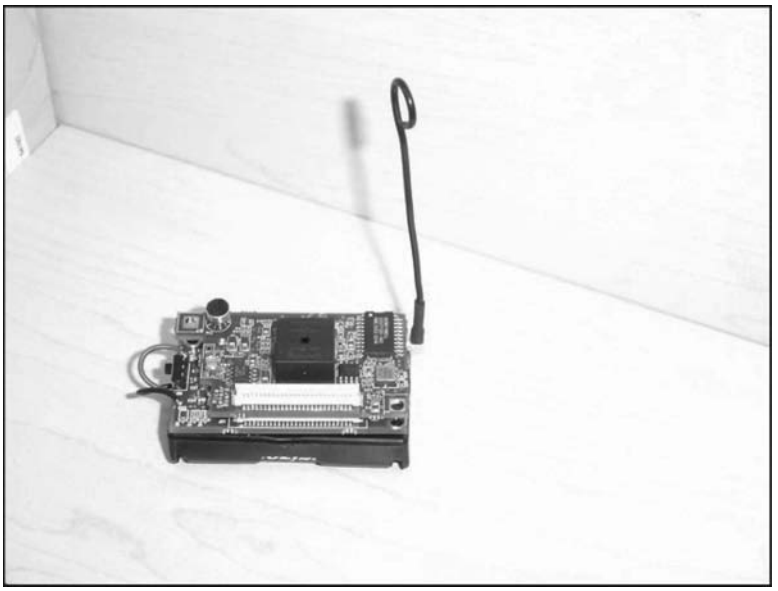


FIGURE 9.1 Wireless sensor node or Mote made by Crossbow Technology, Inc. in San Jose, California.

speed, storage capacity, and communication bandwidth. Additional design challenges include limited power that a Mote can harvest or store, ability to withstand harsh environmental conditions, ability to cope with node failures, mobility of nodes, dynamic network topology, and unattended operation. These constraints require new hardware designs and novel network architectures. Several standards are currently under development for wireless sensor networks. One example is ZigBee, which is a mesh-networking standard intended for uses such as industrial control, embedded sensing, medical data collection, building automation.* Another recent standard is WirelessHART, which is an extension of the HART Protocol for industrial automation.†

From a signal-processing point of view, the main challenge is the distributed fusion of sensor data across the network. This is because individual sensor nodes are often not able to provide useful or comprehensive information about the quantity under observation. Furthermore, due to the variable environmental conditions in which sensor devices may be deployed, one can expect a fraction of the sensor nodes to be malfunctioning. Therefore, the underlying distributed algorithms must be robust with respect to device failures.

In this chapter, we introduce a powerful class of information fusion algorithms, which are based on formulating the sensor fusion problem as a "convex feasibility problem." There are great advantages to formulating a sensor network fusion problem as a convex feasibility problem. The most basic advantage is that a solution can be found in a distributed fashion by using a series of independent projections onto independent convex sets. Another key advantage is that the individual projections can be computed, very reliably and efficiently, using methods for convex optimization [1]. These solution methods are reliable enough to be embedded in real-time system.

* See the ZigBee Alliance Web site at www.zigbee.org
† See the HART Communication Foundation Web site at http://www.hartcomm2.org/index.html

Other desirable features of the convex feasibility approach include the following:

1. Global solution is unique and stable in the sense that small perturbations in the observed data will cause a small change in the solution.
2. Functional form of the solution will depend on the choice of the generalized distance used in the projections. Therefore, a functional form which is easy to manipulate or interpret for a specific application (for instance, a rational function) can be obtained using a proper generalized distance.
3. Formulation can be applied to a variety of network topologies. Some topologies allow for the most efficient computation, some allow for the most robust setup, and others lead to various degrees of compromise between these desirable properties.
4. Formulation has a very rich mathematical structure relying on recent results in several fields of applied mathematics including convex analysis, parallel optimization, and regularization theory.

To maintain clarity and simplicity, we will focus on solving a concrete distributed estimation problem. However, the fusion algorithms that result from our formulations are very general and can be used to solve other sensor network signal-processing problems as well.

9.1.1 Notation

Vectors are denoted by capital letters. Boldface capital letters are used for matrices. Elements of a matrix $\mathbf{A}$ are referred to as $[\mathbf{A}]_{ij}$. We denote the set of real M-tuples by $\mathbb{R}^M$ and use the notation $\mathbb{R}_+$ for positive real numbers. The expected value of a random variable x is denoted by $E\{x\}$. The linear convolution operator is denoted by $\star$. The spaces of Lebesgue-measurable functions are represented by $\mathbf{L}^1(a,b)$, $\mathbf{L}^2(a,b)$, etc. The end of an example is indicated using the symbol $\diamond$.

9.2 Case Study: Spectrum Analysis Using Sensor Networks

9.2.1 Background

A spectrum analyzer or spectral analyzer is a device used to examine the spectral composition of some electrical, acoustic, or optical waveform. Questions such as "Does most of the power of the signal reside at low or high frequencies?" or "Are there resonance peaks in the spectrum?" are often answered as a result of a spectral analysis. Spectral analysis finds frequent and extensive use in many areas of physical sciences. Examples abound in oceanography, electrical engineering, geophysics, astronomy, and hydrology. Throughout this chapter, we will use spectrum analysis as a benchmark signal-processing problem to demonstrate our distributed information fusion algorithms.

Consider the scenario in Figure 9.2, where a sound source (a speaker) is monitored by a collection of Motes put at various known locations in a room . Because of reverberation, noise, and other artifacts, the signal arriving at each Mote location is different. The Motes (which constitute the sensor nodes in our network) are equipped with microphones, sampling devices, sufficient signal-processing hardware and some communication means. Each Mote can process its observed data, come up with some statistical inference about it, and share the result with other nodes in the network. However, to save energy and communication bandwidth, the "Motes are not allowed to share their raw observed data with each other." Now, how should the network operate so that an estimate of the frequency spectrum of the sound source consistent with the observations made by all Motes is obtained?

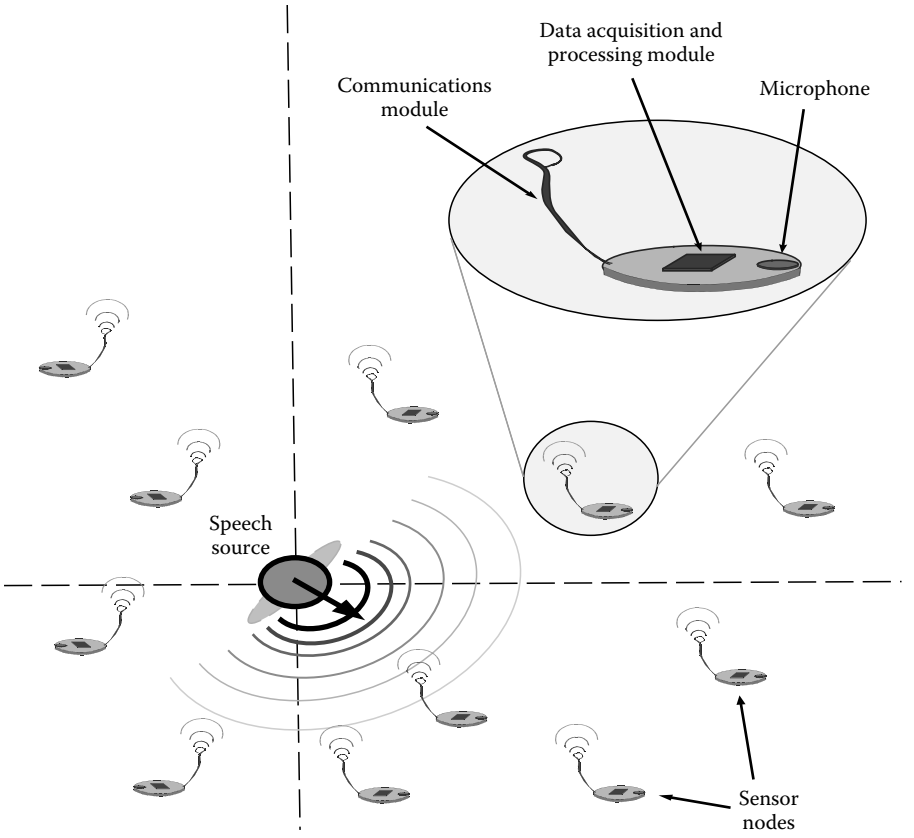


FIGURE 9.2 Sensor network monitoring a stationary sound source in room.

To formulate this problem mathematically, we assume that the sound signal under observation, called $x(n)$, is a random process.* It is well known that a complete statistical description of a zero-mean Gaussian wide-sense stationary (WSS) random process is provided by its "autocorrelation sequence" (ACS)

$$R_x(k) \triangleq E\{x(n)x(n+k)\}$$

or, equivalently, by its "power spectrum" also known as "power spectral density"

$$P_x(e^{j\omega}) = \sum_{k=-\infty}^{\infty} R_x(k)e^{-j\omega k}.$$

The ACS is a time-domain description of the second-order statistics of a random process. The power spectrum provides a frequency domain description of the same statistics. Here, we are concerned with determining the "power spectrum" of a random signal using distributed data obtained by a sensor network.†

* The reader is referred to the excellent texts [2–5] for basic introduction to random processes.

† The problem of estimating the power spectrum of a random signal when the signal itself is not available but some measured signals derived from it are observable has been studied in [6]. The approach developed in [6], however, leads to a centralized fusion algorithm, which is not suited to sensor network applications.

9.2.2 Estimating the Power Spectrum of a Signal Source Using Sensor Network Data

Again, let $x(n)$ denote a discrete version of the signal produced by the source and assume that it is a zero-mean Gaussian WSS random process. The sampling frequency f_s associated with $x(n)$ is arbitrary and depends on the frequency resolution desired in the spectrum estimation process.

We denote by $v_i(n)$ the signal produced at the front end of the ith sensor node. We assume that $v_i(n)$ are related to the original source signal $x(n)$ by the model shown in Figure 9.3. The linear filter $H_i(z)$ in this figure models the combined effect of room reverberations, microphone's frequency response, and an additional filter which the system designer might want to include. The decimator block which follows the filer represents the (potential) difference between the sampling frequency f_s associated with $x(n)$ and the actual sampling frequency of the Mote's sampling device. Here, it is assumed that the sampling frequency associated with $v_i(n)$ is f_s/N_i where N_i is a fixed natural number.

It is straightforward to show that the signal $v_i(n)$ in Figure 9.3 is also a WSS processes. The autocorrelation coefficients $R_{v_i}(k)$ associated with $v_i(n)$ are given by

$$R_{v_i}(k) = R_{x_i}(N_i k) \tag{9.1}$$

and

$$R_{x_i}(k) = (h_i(k) \star h_i(-k)) \star R_x(k), \tag{9.2}$$

where $h_i(k)$ denotes the impulse response of $H_i(z)$. We can express $R_{v_i}(k)$ as a function of the source signal's power spectrum as well. To do this, we define $G_i(z) \triangleq H_i(z)H_i(z^{-1})$ and then use it to write Equation 9.2 in the frequency domain:

$$R_{x_i}(k) = \frac{1}{2\pi}\int_{-\pi}^{\pi} P_x(e^{j\omega})G_i(e^{j\omega})e^{jk\omega}d\omega. \tag{9.3}$$

Combining Equations 9.1 and 9.3, we then get

$$R_{v_i}(k) = \frac{1}{2\pi}\int_{-\pi}^{\pi} P_x(e^{j\omega})G_i(e^{j\omega})e^{jN_i k\omega}d\omega. \tag{9.4}$$

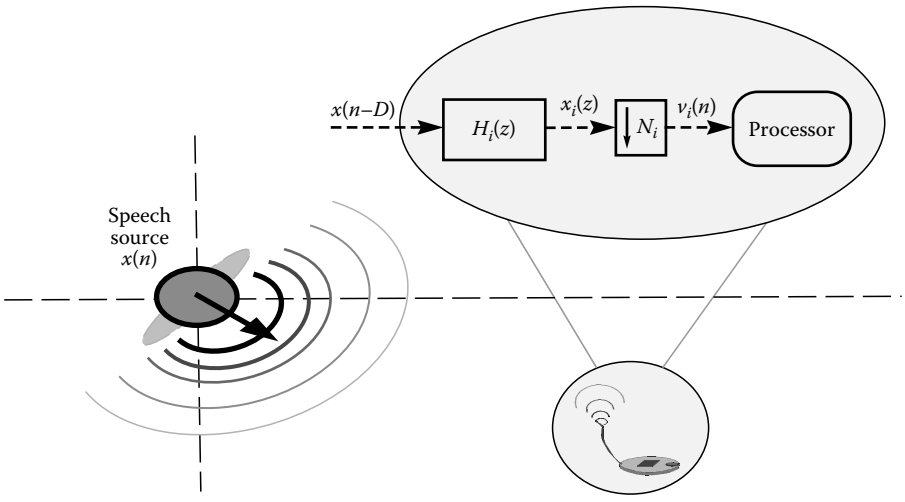


FIGURE 9.3 Relation between the signal $v_i(n)$ produced by the front end of the ith sensor and the original source signal $x(n)$.

The above formula shows that $P_x(e^{j\omega})$ uniquely specifies $R_{v_i}(k)$ for all values of k. However, the reverse is not true. That is, in general, knowing $R_{v_i}(k)$ for some or all values of k is not sufficient for characterizing $P_x(e^{j\omega})$ uniquely.

Recall that $v_i(n)$ is a WSS signal so all the statistical information that can be gained about it is confined in its autocorrelation coefficients. One might use the signal-processing hardware available at each sensor node and estimate the autocorrelation coefficients $R_{v_i}(k)$ for some k, say $0 \le k \le L-1$. This leads us to pose the sensor network spectrum estimation problem as follows:

PROBLEM 9.1 *Let $\mathcal{Q}_{i,k}$ denote the set of all power spectra which are consistent with the kth autocorrelation coefficient $R_{v_i}(k)$ estimated at the ith sensor node. That is, $P_x(e^{j\omega}) \in \mathcal{Q}_{i,k}$ if*

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} P_x(e^{j\omega})G_i(e^{j\omega})e^{jMk\omega}d\omega = R_{v_i}(k),$$

$$P_x(e^{j\omega}) \ge 0,$$

$$P_x(e^{j\omega}) = P_x(e^{-j\omega}),$$

$$P_x(e^{j\omega}) \in \mathbf{L}^1(-\pi, \pi).$$

Define $\mathcal{Q} \triangleq \bigcap_{i=1}^{N}\bigcap_{k=0}^{L-1}\mathcal{Q}_{i,k}$ where N is the number of nodes in the network and L is the number of autocorrelation coefficients estimated at each node. Find a $P_x(e^{j\omega})$ in $\mathcal{Q}$.

If we ignore measurement imperfections and assume that the observed autocorrelation coefficients $R_{v_i}(k)$ are exact, then the sets $\mathcal{Q}_{i,k}$ are nonempty and admit a nonempty intersection $\mathcal{Q}$ as well. In this case, $\mathcal{Q}$ contains infinitely many $P_x(e^{j\omega})$. When the measurements $v_i(n)$ are contaminated by noise or $R_{v_i}(k)$ are estimated based on finite-length data records, the intersection set $\mathcal{Q}$ might be empty due to the potential inconsistency of the autocorrelation coefficients estimated by different sensors. Thus, Problem 9.1 has either no solution or infinitely many solutions. Problems which have such undesirable properties are called "ill-posed". Ill-posed problems are studied in the next section.

9.3 Inverse and Ill-Posed Problems

The study of inverse problems has been one of the fastest-growing areas in applied mathematics in the last two decades. This growth has largely been driven by the needs of applications in both natural sciences (e.g., inverse scattering theory, astronomical image restoration, and statistical learning theory) and industry (e.g., computerized tomography and remote sensing). The reader is referred to [7–10] for detailed treatments of the theory of ill-posed problems and to [11,12] for applications in inverse scattering and statistical inference, respectively.

The definition, inverse problems are concerned with determining causes for a desired or an observed effect. Most often, inverse problems are much more difficult to deal with (from a mathematical point of view) than their direct counterparts. This is because they might not have a solution in the strict sense or solutions might not be unique or depend on data continuously. Mathematical problems having such undesirable properties are called "ill-posed problems" and cause severe numerical difficulties (mostly because of the discontinuous dependence of solutions on the data).

Formally, a problem of mathematical physics is called "well-posed or well-posed in the sense of Hadamard" if it fulfills the following conditions:

1. For all admissible data, a solution exists.
2. For all admissible data, the solution is unique.
3. The solution depends continuously on the data.

A problem for which one or more of the above conditions are violated is called ill-posed. Note that the conditions mentioned above do not make a precise definition for well-posedness. To make a precise definition in a concrete situation, one has to specify the notion of a solution, which data are considered admissible, and which topology is used for measuring continuity.

If a problem is well-posed, then it stands a good chance of solution on a computer using a stable algorithm. If it is not well-posed, it needs to be reformulated for numerical treatment. Typically, this involves including additional assumptions, such as smoothness of solution. This process is known as "regularization." The theory of regularization is well developed for linear inverse problems and will be introduced in Section 9.3.2.

9.3.1 Ill-Posed Linear Operator Equations

Consider the linear operator equation

$$Ax = y \tag{9.5}$$

defined by the continuous operator A that maps the elements x of a metric space $\mathcal{E}_1$ into elements y of the metric space $\mathcal{E}_2$. In the early 1900s, noted French mathematician Jacques Hadamard observed that under some (very general) circumstances the problem of solving the operator Equation 9.5 is ill-posed. This is because, even if there exists a unique solution $x \in \mathcal{E}_1$ that satisfies the equality 9.5, a small deviation on the right-hand side can cause large deviations in the solution. The following example illustrates this issue.

Example 9.1

Let A denote a Fredholm integral operator of the first kind. Thus, we define

$$(Ax)(s) \stackrel{\triangle}{=} \int_a^b K(s,t)x(t)\mathrm{d}t. \tag{9.6}$$

The kernel $K(s,t)$ is continuous on $[a\ b] \times [a\ b]$ and maps a function $x(t)$ continuous on $[a\ b]$ to a function $y(s)$ also continuous on $[a\ b]$. We observe that the continuous function

$$g_\omega(s) \stackrel{\triangle}{=} \int_a^b K(s,t)\sin(\omega t)\mathrm{d}t, \tag{9.7}$$

which is formed by means of the kernel $K(s,t)$ possesses the property

$$\lim_{\omega\to\infty} g_\omega(s) = 0, \ \ for\ every\ s \in [a,b]. \tag{9.8}$$

The above property is a consequence of the fact that the Fourier series coefficients of a continuous function tend to zero at high frequencies. See, for example, ([13], *Chapter 14, Section I*). *Now, consider the integral equation*

$$Ax = y + g_\omega, \tag{9.9}$$

where y is given and g_ω is defined in Equation 9.7. As the above equation is linear, it follows using Equation 9.7 that its solution $\hat{x}(t)$ has the form

$$\hat{x}(t) = x^*(t) + \sin(\omega t), \tag{9.10}$$

where $x^*(t)$ is a solution to the original integral equation $Ax = y$. For sufficiently large ω, the right-hand side of Equation 9.9 differs from the right-hand side of Equation 9.5 only by the small *amount $g_\omega(s)$, while its solution differs from that of Equation 9.5 by the amount $\sin(\omega t)$. Thus, the problem of solving Equation 9.5 where A is a Fredholm integral operator of the first kind is ill-posed.* ♢

One can easily verify that the problem of solving the operator Equation 9.5 is equivalent to finding an element $x^* \in \mathcal{E}_1$ such that the functional

$$R(x) \triangleq \|Ax - y\|_{\mathcal{E}_2} \tag{9.11}$$

is minimized.* Note that the minimizing element $x^* \in \mathcal{E}_1$ always exists even when the original Equation 9.5 does not have a solution. In any case, if the right-hand side of Equation 9.5 is not exact, that is, if we replace y by y_δ such that $\|y - y_\delta\|_{\mathcal{E}_2} < \delta$ where δ is a small value, a new element $x_\delta \in \mathcal{E}_1$ will minimize the functional

$$R_\delta(x) \triangleq \|Ax - y_\delta\|_{\mathcal{E}_2}. \tag{9.12}$$

However, the new solution x_δ is not necessarily close to the first solution x^* even if δ tends to zero. In other words, $\lim_{\delta \to 0} \|x^* - x_\delta\|_{\mathcal{E}_1} \neq 0$ when the operator equation $Ax = y$ is ill-posed.

9.3.2 Regularization Methods for Solving Ill-Posed Linear Operator Equations

Hadamard [14] thought that ill-posed problems are a pure mathematical phenomenon and that all real-life problems are well-posed. However, in the second half of the 20th century, a number of very important real-life problems were found to be ill-posed. In particular, as we just discussed, ill-posed problems arise when one tries to reverse the cause–effect relations to find unknown causes from known consequences. Even if the cause–effect relationship forms a one-to-one mapping, the problem of inverting it can be ill-posed. The discovery of various "regularization methods" by Tikhonov, Ivanov, and Phillips in the early 1960s made it possible to construct a sequence of "well-posed solutions" that converges to the desired one.

Regularization theory was one of the first signs of existence of "intelligent inference." It demonstrated that whereas the "self-evident" methods of solving an operator equation might not work, the "non-self-evident" methods of regularization theory do. The influence of the philosophy created by the theory of regularization is very deep. Both the regularization philosophy and the regularization techniques became widely disseminated in many areas of science and engineering [9,10].

9.3.2.1 Tikhonov's Method

In the early 1960s, it was discovered by Tikhonov [15,16] that if instead of the functional $R_\delta(x)$ one minimizes

$$R_{\text{reg}}(x) \triangleq \|Ax - y_\delta\|_{\mathcal{E}_2} + \xi(\delta)S(x), \tag{9.13}$$

where $S(x)$ is a "stabilizing functional" (that belongs to a certain class of functionals) and $\xi(\delta)$ is an appropriately chosen constant (whose value depends on the "noise" level δ), then one obtains a sequence of solutions x_δ that converges to the desired one as δ tends to zero. For the above result to be valid, it is required that

1. The problem of minimizing $R_{\text{reg}}(x)$ be well-posed for fixed values of δ and $\xi(\delta)$.
2. $\lim_{\delta \to 0} \|x^* - x_\delta\|_{\mathcal{E}_1} \to 0$ when $\xi(\delta)$ is chosen appropriately.

---

* To save in notation, we write $\|a - b\|_{\mathcal{E}}$ to denote the "distance" between the two elements $a, b \in \mathcal{E}$ whether the metric space $\mathcal{E}$ is a normed space or not. If $\mathcal{E}$ is a normed space too, our notation is self-evident. Otherwise, it should be interpreted only as a "symbol" for the distance between a and b.

Consider a real-valued lower semi-continuous* functional $S(x)$. We shall call $S(x)$ a "stabilizing functional" if it possesses the following properties:

1. Solution of the operator equation $Ax = y$ belongs to the domain of definition $\mathcal{D}(S)$ of the functional S.
2. $S(x) \geq 0, \quad \forall x \in \mathcal{D}(S)$.
3. Level sets $\{x : S(x) \leq c\}$, $c = const.$, are all compact.

It turns out that the above conditions are sufficient for the problem of minimizing $R_{\text{reg}}(x)$ to be well-posed [7, p. 51]. Now, the important remaining problem is to determine the functional relationship between δ and $\xi(\delta)$ such that the sequence of solutions obtained by minimizing Equation 9.13 converges to the solution of Equation 9.11 as δ tends to zero. The following theorem establishes sufficient conditions on such a relationship:

THEOREM 9.1 [12, p. 55] *Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be two metric spaces and let $A : \mathcal{E}_1 \to \mathcal{E}_2$ be a continuous and one-to-one operator. Suppose that for $y \in \mathcal{E}_2$ there exists a solution $x \in \mathcal{D}(S) \subset \mathcal{E}_1$ to the operator equation $Ax = y$. Let y_δ be an element in $\mathcal{E}_2$ such that $\|y - y_\delta\|_{\mathcal{E}_2} \leq \delta$. If the parameter $\xi(\delta)$ is chosen such that*

(i) $\xi(\delta) \to 0$ *when* $\delta \to 0$

(ii) $\lim_{\delta \to 0} \dfrac{\delta^2}{\xi(\delta)} < \infty$

Then the elements $x_\delta \in \mathcal{D}(S)$ minimizing the functional

$$R_{reg}(x) = \|Ax - y_\delta\|_{\mathcal{E}_2} + \xi(\delta) S(x)$$

converge to the exact solution x as $\delta \to 0$.

If $\mathcal{E}_1$ is a Hilbert space, the stabilizing functional $S(x)$ may simply be chosen as $\|x\|^2$, which, indeed, is the original choice made by Tikhonov. In this case, the level sets of $S(x)$ will only be weakly compact. However, the convergence of the regularized solutions will be a strong one in view of the properties of Hilbert spaces. The conditions imposed on the parameter $\xi(\delta)$ are, nevertheless, more stringent than those stated in the above theorem.†

9.3.2.2 Residual Method

The results presented above are fundamentals in Tikhonov's theory of regularization. Tikhonov's theory, however, is only one of several proposed schemes for solving ill-posed problems. An important variation known as Residual Method was introduced by Phillips [17]. In Phillips' method, one minimize the functional

$$R_P(x) \stackrel{\triangle}{=} S(x)$$

subject to the constraint

$$\|Ax - y_\delta\|_{\mathcal{E}_2} \leq \mu,$$

where μ is a fixed constant. The stabilizing functional $S(x)$ is defined as in the previous subsection.

---

* A function $f : \mathbb{R}^N \to [-\infty, \infty]$ is called lower semi-continuous at $X \in \mathbb{R}^N$ if for any $t < f(X)$ there exists $\delta > 0$ such that for all $y \in \mathcal{B}(X, \delta)$, $t < \delta$. The notation $\mathcal{B}(X, \delta)$ represents a ball with center at X and radius δ. This definition generalizes to functional spaces by using the appropriate metric in defining $\mathcal{B}(X, \delta)$.

† In this case, $\xi(\delta)$ should converge to zero "strictly slower" than δ^2. In more precise terms, $\lim_{\delta \to 0} \frac{\delta^2}{\xi(\delta)} = 0$ must hold.

9.3.2.3 Quasi-Solution Method

The Quasi-Solution Method was developed by Ivanov [18,19]. In this method, one minimizes the functional

$$R_I(x) \triangleq \|Ax - y_\delta\|_{\mathcal{E}_2}$$

subject to the constraint

$$S(x) \leq \sigma,$$

where σ is a fixed constant. Again, the stabilizing functional $S(x)$ is defined as in Tikhonov's method.

Note that the three regularization methods mentioned above contain one free parameter (ξ in Tikhonov's method, μ for Phillips' method, and σ in Ivanov's method). It has been shown [20] that these methods are all equivalent in the sense that if one of the methods (say Phillips') for a given value of its parameter (say μ^*) produces a solution x^*, then there exist corresponding values of parameters of the other two methods that produce the same solution. We remark in passing that a smart choice of the free parameter is crucial in obtaining a good (fast converging) solution using any of the regularization methods mentioned above. There exist several principles for choosing the free parameter in an optimal fashion [9, Section 4.3, 10, Chapter 2].

9.4 Spectrum Estimation Using Generalized Projections

The sensor network spectrum estimation problem (Problem 9.1) posed in Section 9.2.2 is essentially finding a $P(e^{j\omega})$ in the intersection of the feasible sets $\mathcal{Q}_{i,k}$. It is easy to verify that the sets $\mathcal{Q}_{i,k}$ are closed and convex [6]. The problem of finding a point in the intersection of finitely many closed convex sets is known as the convex feasibility problem and is an active area of research in applied mathematics.

An elegant way to solve a convex feasibility problem is to employ a series of "generalized projections" [21]. A generalized projection is essentially a regularization method with a "generalized distance" serving as the stabilizing functional. A great advantage of using the generalized projections formulation is that the solution $P^* \in \mathcal{Q}$ can be found using a series of projections onto the intermediate sets $\mathcal{Q}_{i,k}$. These intermediate projections can be computed locally at each sensor node thus allowing the computations to be done simultaneously and in a highly distributed fashion.

A generalized distance is a real-valued nonnegative function of two vector variable $D(X, Y)$ defined in a specific way such that its value may represent the distance between X and Y in some generalized sense. When defining generalized distances, it is customary not to require the symmetry condition. Thus, $D(X, Y)$ may not be the same as $D(Y, X)$. Moreover, we do not insist on the triangle inequality that a traditional metric must obey either.

Example 9.2

Let $P_1(e^{j\omega}) > 0$ *and* $P_2(e^{j\omega}) > 0$ *be two power spectra in* $\mathbf{L}^1(-\pi, \pi)$. *The functions*

$$D_1(P_1, P_2) = \int_{-\pi}^{\pi} (P_1 - P_2)^2 \, d\omega$$

$$D_2(P_1, P_2) = \int_{-\pi}^{\pi} \left(P_1 \ln \frac{P_1}{P_2} + P_2 - P_1 \right) d\omega$$

$$D_3(P_1, P_2) = \int_{-\pi}^{\pi} \left(\frac{P_1}{P_2} - \ln \frac{P_1}{P_2} - 1 \right) d\omega$$

can be used to measure the generalized distance between $P_1(e^{j\omega})$ and $P_2(e^{j\omega})$. These functions are nonnegative and become zero if and only if $P_1 = P_2$. Note that D_1 is simply the Euclidean distance between P_1 and P_2. The functions D_2 and D_3 have roots in information theory and statistics. They are known as the Kullback–Leibler divergence and Burg cross entropy, respectively. ◇

By using a suitable generalized distance, we can convert our original sensor network spectrum estimation problem (Problem 9.1) into the following minimization problem:

PROBLEM 9.2 *Let $\mathcal{Q}$ be defined as in Problem 9.1. Find $P_x^*(e^{j\omega})$ in $\mathcal{Q}$ such that*

$$P^* = \arg\min_{P\in\mathcal{Q}} D(P, P_0), \tag{9.14}$$

where $P_0(e^{j\omega})$ is an arbitrary power spectrum, say $P_0(e^{j\omega}) = 1, -\pi \le \omega < \pi$.

When a unique P^* exists, it is called the generalized projection of P_0 onto $\mathcal{Q}$ [22]. In general, a projection of a given point onto a convex set is defined as another point, which has two properties: First, it belongs to the set onto which the projection operation is performed and, second, it renders a minimal value to the distance between the given point and any point of the set (Figure 9.4).

If the Euclidean distance, $||X - Y||$ is used in this context then the projection is called a metric projection. In some cases, such as the spectrum estimation problem considered here, it turns out to be very useful to introduce more general means to measure the distance between two vectors. The main reason is that the functional form of the solution will depend on the choice of the distance measure used in the projection. Often, a functional form which is easy to manipulate or interpret (for instance, a rational function) cannot be obtained using the conventional Euclidean metric.

It can be shown that the distances D_1 and D_2 in Example 9.2 lead to well-posed solutions for P^*. The choice D_3 will lead to a unique solution given that certain singular power spectra are excluded

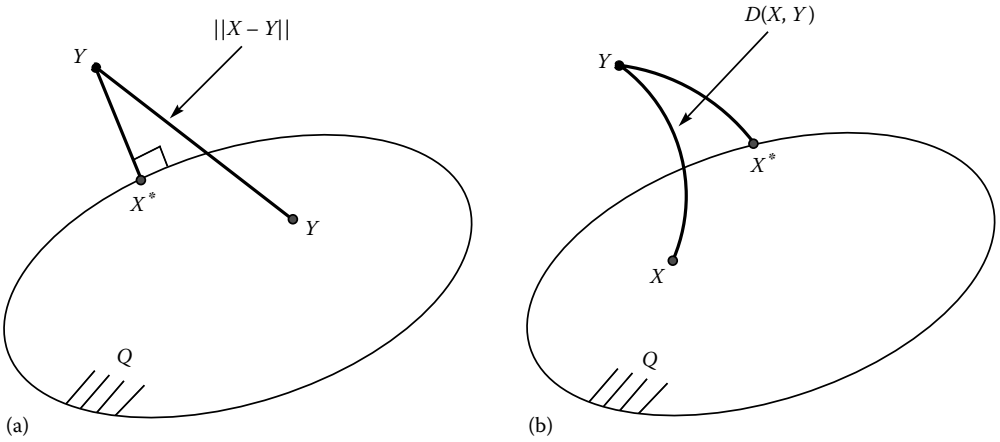


FIGURE 9.4 (a) Symbolic depiction of metric projection and (b) generalized projection of a vector Y into a closed convex set $\mathcal{Q}$. In (a) the projection X^* is selected by minimizing the metric $||X - Y||$ over all $X \in \mathcal{Q}$ while in (b) X^* is found by minimizing the generalized distance $D(X, Y)$ over the same set.

from the space of valid solutions [23]. It is not known whether D_3 will lead to a stable solution. As a result, the well-posedness of Problem 9.2 when D_3 is used is not yet established.*

9.5 Distributed Algorithms for Calculating Generalized Projection

As we mentioned before, a very interesting aspect of the generalized projections formulation is that the solution $P^* \in \mathcal{Q}$ can be found using a series of projections onto the intermediate sets $\mathcal{Q}_{i,k}$. In this section, we first calculate the generalized projection of a given power spectrum onto the sets $\mathcal{Q}_{i,k}$ for the sample distance functions introduced in Example 9.2. Then, we propose a distributed algorithm for calculating the final solution P^* from these intermediate projections.

Let $P_{[P_1 \mapsto \mathcal{Q}_{i,k}; D_j]}$ denote the power spectrum resulting from projecting a given power spectrum P_1 onto the set $\mathcal{Q}_{i,k}$ using a given distance functions D_j. That is,

$$P_{[P_1 \mapsto \mathcal{Q}_{i,k}; D_j]} \triangleq \arg \min_{P \in \mathcal{Q}_{i,k}} D_j(P, P_1). \tag{9.15}$$

Using standard techniques from calculus of variations, we can show that the generalized distances D_1, D_2, and D_3 introduced in Example 9.2 result in projections of the form

$$\begin{aligned}
P_{[P_1 \mapsto \mathcal{Q}_{i,k}; D_1]} &= P_1(e^{j\omega}) - \alpha G_i(e^{j\omega}) \cos(Mk\omega), \\
P_{[P_1 \mapsto \mathcal{Q}_{i,k}; D_2]} &= P_1(e^{j\omega}) \exp\left(-\beta G_i(e^{j\omega}) \cos(Mk\omega)\right), \\
P_{[P_1 \mapsto \mathcal{Q}_{i,k}; D_3]} &= \left(P_1(e^{j\omega})^{-1} + \gamma G_i(e^{j\omega}) \cos(Mk\omega)\right)^{-1},
\end{aligned}$$

where α, β, and γ are parameters (Lagrange multipliers). These parameter should be chosen such that in each case $P_{[P_1 \mapsto \mathcal{Q}_{i,k}; D_j]} \in \mathcal{Q}_{i,k}$. That is,

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} P_{[P_1 \mapsto \mathcal{Q}_{i,k}; D_j]} G_i(e^{j\omega}) e^{jMk\omega} d\omega = R_{v_i}(k). \tag{9.16}$$

The reader may observe that the above equation leads to a closed-form formula for α but in general finding β and γ requires numerical methods. The projection formulae developed above can be employed in a variety of iterative algorithms to fined a solution in the intersection of $\mathcal{Q}_{i,k}$. We discuss two example algorithms below.

9.5.1 Ring Algorithm

Ring Algorithm is a very simple algorithm: it starts with an initial guess $P^{(0)}$ for $P_x(e^{j\omega})$ and then calculates a series of successive projections onto the constraint sets $\mathcal{Q}_{i,k}$. Then, it takes the last projection, now called $P^{(1)}$, and projects it back onto the first constraint set. Continuing this process will generate a sequence of solutions $P^{(0)}, P^{(1)}, P^{(2)}, \ldots$ which will eventually converge to a solution $P^* \in \bigcap_{i,k} \mathcal{Q}_{i,k}$ [21]. Steps of the Ring Algorithm are summarized in the text box below. A graphical representation of this algorithm is shown in Figure 9.5.

* Well-posedness of the minimization problem (Equation 9.14) when D is the Kullback–Leibler divergence D_2 has been established in several works including [24–28]. Well-posedness results exist for certain classes of generalized distance functions as well [28,29]. Unfortunately, the Burg cross entropy D_3 does not belong to any of these classes. While Burg cross entropy lacks theoretical support as a regularizing functional, it has been used successfully to resolve ill-posed problems in several applications including spectral estimation and image restoration (see, for example, [30] and references therein). The desirable feature of Burg cross entropy in the context of spectrum estimation is that its minimization (subject to linear constraints $P_x^*(e^{j\omega}) \in \mathcal{Q}$) leads to rational power spectra.

Ring Algorithm

Input: A distance function $D_j(P_1, P_2)$, an initial power spectrum $P_0(e^{j\omega})$, the squared sensor frequency responses $G_i(e^{j\omega})$, and the autocorrelation estimates $R_{v_i}(k)$ for $k = 0, 1, \ldots, L-1$ and $i = 1, 2, \ldots, N$.
Output: A power spectrum $P_*(e^{j\omega})$.

Procedure:

1. Let $m = 0$, $i = 1$, and $P^{(m)} = P_0$.
2. Send $P^{(m)}$ to the ith sensor node.

   At the ith sensor:

   (i) Let $k = 0$ and define $\tilde{P}_k = P^{(m)}$.

   (ii) Calculate $\tilde{P}_k = P_{[\tilde{P}_{k-1} \mapsto \mathcal{Q}_{i,k}; D_j]}$ for $k = 1, 2, \ldots, L-1$.

   (iii) If $D(\tilde{P}_{L-1}, \tilde{P}_0) > \epsilon$ then let $\tilde{P}_0 = \tilde{P}_{L-1}$ and go back to item (ii). Otherwise, let $i = i + 1$ and go to Step 3.
3. If $(i \ mod \ N) = 1$ then set $m = m + 1$ and reset i to 1. Otherwise, set $P^{(m)} = \tilde{P}_{L-1}$ and go back to Step 2.
4. Define $P^{(m)} = \tilde{P}_{L-1}$. If $D(P^{(m)}, P^{(m-1)}) > \epsilon$, go back to Step 2. Otherwise output $P^* = P^{(m)}$ and stop.

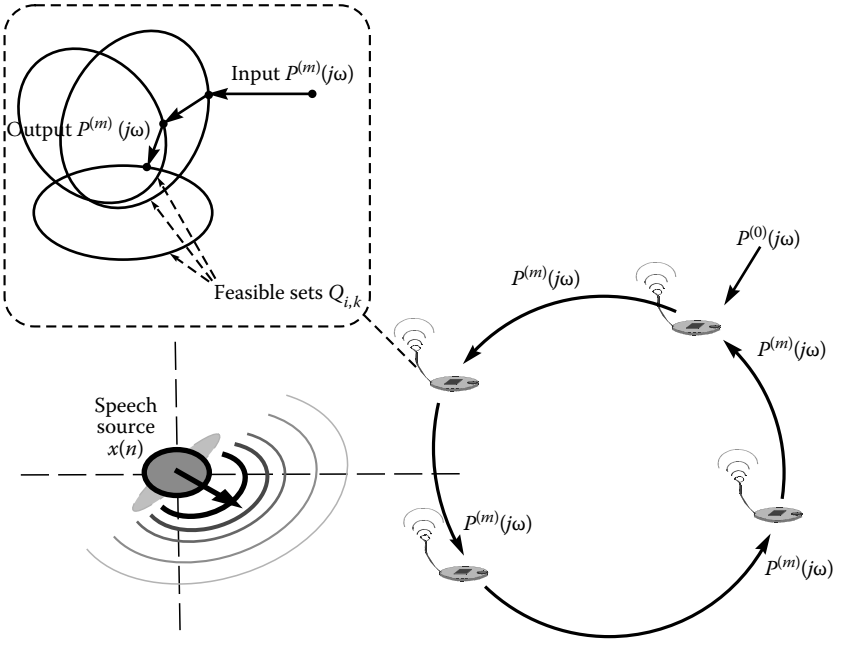


FIGURE 9.5 Graphical depiction of the Ring Algorithm. For illustrative reasons, only three feasible sets $Q_{i,k}$ are shown in the inside picture. Also, it is shown that the output spectrum $P^{(m)}(e^{j\omega})$ is obtained from the input $P^{(m)}(e^{j\omega})$ only after three projections. In practice, each sensor node has L feasible sets and has to repeat the sequence of projections many times before it can successfully project the input $P^{(m)}(e^{j\omega})$ into the intersection of its feasible sets.

Example 9.3

Consider a simple 4-sensor network similar to the one shown in Figure 9.5. Assume that the down-sampling ratio in each Mote is equal to 4. Thus, $N_0 = N_1 = N_2 = N_3 = 4$. *Assume, further, that the transfer functions* $H_0(z)$ *to* $H_3(z)$ *which relate the Motes' front-end output* $v_i(n)$ *to the original source signal* $x(n)$ *are given as follows:*

$$H_0(z) = \frac{0.0753 + 0.1656z^{-1} + 0.2053z^{-2} + 0.1659z^{-3} + 0.0751z^{-4}}{1.0000 - 0.8877z^{-1} + 0.6738z^{-2} - 0.1206z^{-3} + 0.0225z^{-4}}$$

$$H_1(z) = \frac{0.4652 - 0.1254z^{-1} - 0.3151z^{-2} + 0.0975z^{-3} - 0.0259z^{-4}}{1.0000 - 0.6855z^{-1} + 0.3297z^{-2} - 0.0309z^{-3} + 0.0032z^{-4}}$$

$$H_2(z) = \frac{0.3732 - 0.8648z^{-1} + 0.7139z^{-2} - 0.1856z^{-3} - 0.0015z^{-4}}{1.0000 - 0.5800z^{-1} + 0.5292z^{-2} - 0.0163z^{-3} + 0.0107z^{-4}}$$

$$H_3(z) = \frac{0.1931 - 0.4226z^{-1} + 0.3668z^{-2} - 0.0974z^{-3} - 0.0405z^{-4}}{1.0000 + 0.2814z^{-1} + 0.3739z^{-2} + 0.0345z^{-3} - 0.0196z^{-4}}$$

The above transfer functions were chosen to show typical low-pass, band-pass, and high-pass characteristics (Figure 9.6). They were obtained using standard filter design techniques. The input signal whose power spectrum is to be estimated was chosen to have a smooth low-pass spectrum . We used the Ring Algorithm with $L = 4$ *and the Euclidean metric* D_1 *as the distance function to estimate the input signal's spectrum. The results are shown in (Figure 9.7). As seen in this figure, the algorithm converges to a solution which is in this case almost identical to the actual input spectrum in less than 100 rounds.* ◊

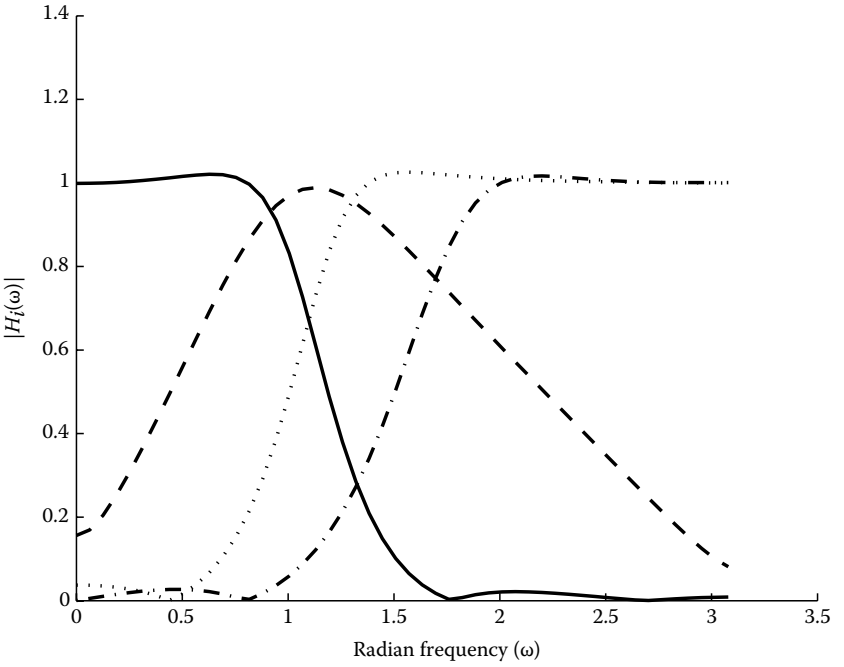


FIGURE 9.6 Frequency response amplitude of the transfer functions used in Example 9.3. The curves show, from left to right, $|H_0(e^{j\omega})|$, $|H_1(e^{j\omega})|$, $|H_2(e^{j\omega})|$, and $|H_3(e^{j\omega})|$.

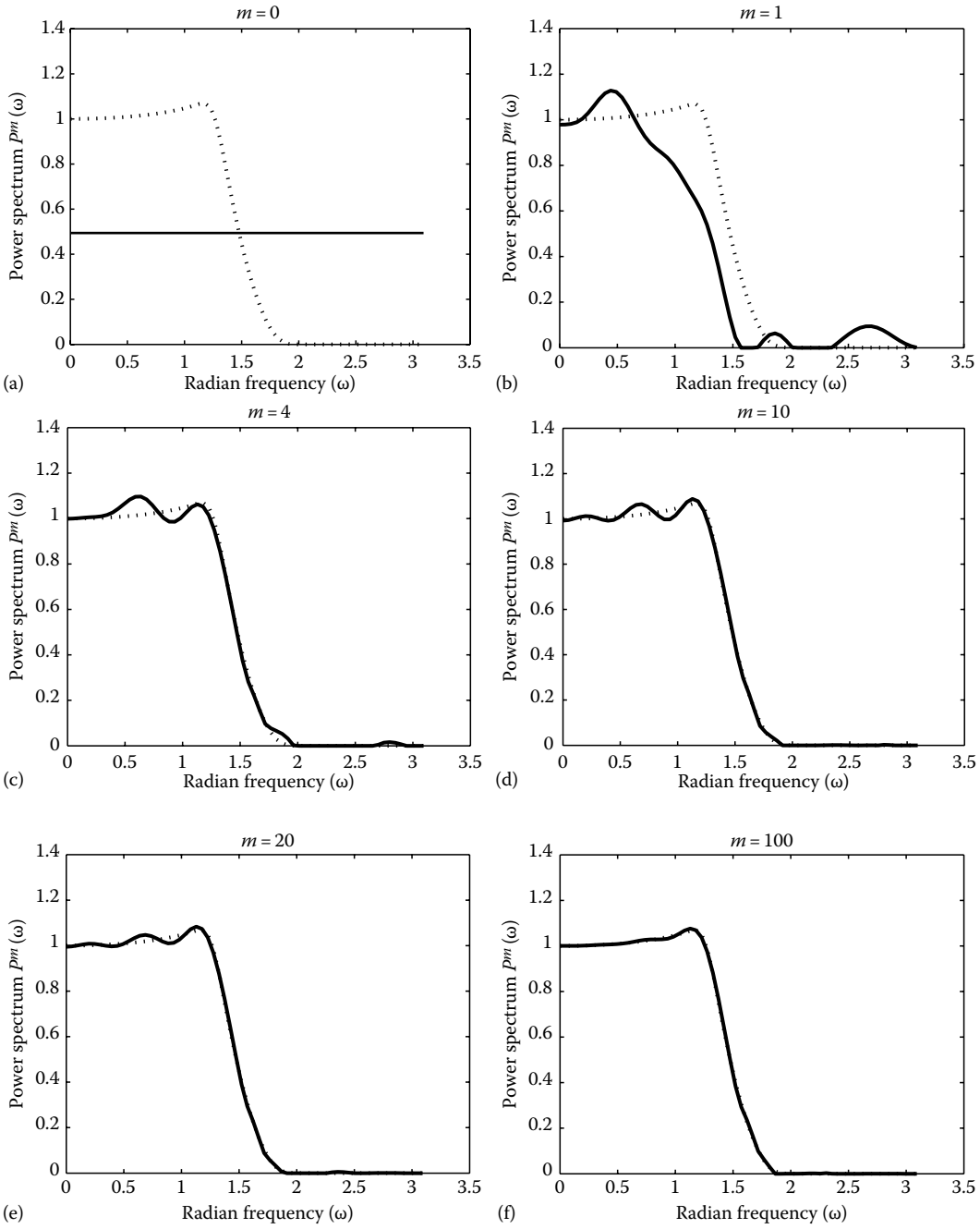


FIGURE 9.7 Ring Algorithm convergence results. In each figure, the dashed curve shows the source signal's actual power spectrum while the solid curve is the estimate obtained by the Ring Algorithm after m rounds. A "round" means projections have been passed through all the nodes in the network.

9.5.2 Star Algorithm

The Ring Algorithm is completely decentralized (Figure 9.8). However, it will not converge to a solution if the feasible sets $\mathcal{Q}_{i,k}$ do not have an intersection (which can happen due to measurement noise) or one or more sensors in the network are faulty. The Star Algorithm is an alternative distributed algorithm for fusing individual sensors' data. It combines successive projections onto $\mathcal{Q}_{i,k}$ with a kind of averaging operation to generate a sequence of solutions $P^{(m)}$. This sequence will eventually converge to a solution $P^* \in \bigcap_{i,k} \mathcal{Q}_{i,k}$ if one exists. The Star Algorithm is fully parallel and hence much faster than the Ring Algorithm. It provides some degree of robustness to individual node's failure as well. However, it includes a centralized step, which needs to be accommodated for when the system's network protocol is being designed. Steps of the Star Algorithm are summarized in the text box below. A graphical representation of this algorithm is shown in Figure 9.10.

Star Algorithm

Input: A distance function $D_j(P_1, P_2)$, an initial power spectrum $P_0(e^{j\omega})$, the squared sensor frequency responses $G_i(e^{j\omega})$, and the autocorrelation estimates $R_{v_i}(k)$.
Output: A power spectrum $P_*(e^{j\omega})$.
Procedure:

1. Let $m = 0$ and $P^{(0)} = P_0$.
2. Send $P^{(m)}$ to all sensor nodes.

   At the ith sensor:

   (i) Let $n = 0$ and define $\tilde{P}^{(n)} = P^{(m)}$.

   (ii) Calculate $\tilde{P}_k = P_{[\tilde{P}^{(n)} \mapsto \mathcal{Q}_{i,k}; D_j]}$ for all k.

   (iii) Calculate $\tilde{P}^{(n+1)} = \arg\min_P \sum_k D(P, \tilde{P}_k)$.

   (iv) If $D(\tilde{P}^{(n+1)}, \tilde{P}^{(n)}) > \epsilon$ go to item (ii) and repeat. Otherwise, define $P_i^{(m)} = \tilde{P}^{(n+1)}$ and send it to the central unit.

2. Receive $P_i^{(m)}$ from all sensor and calculate $P^{(m+1)} = \arg\min_P \sum_i D(P, P_i^{(m)})$.
3. If $D(P^{(m+1)}, P^{(m)}) > \epsilon$, go to step 2 and repeat. Otherwise stop and output $P^* = P^{(m+1)}$.

Example 9.4

Consider a simple 5-sensor network similar to the one shown in Figure 9.10. Assume that the downsampling ratio in each Mote is equal to 4. Thus, again, $N_0 = N_1 = N_2 = N_3 = 4$. Assume, further, that the transfer functions $H_0(z)$ to $H_3(z)$ which relate the Motes' front-end output $v_i(n)$ to the original source signal $x(n)$ are the same as those introduced in Example 9.3. We simulated the Star Algorithm with $L = 4$ and the Euclidean metric D_1 as the distance function to estimate the input signal's spectrum. The results are shown in (Figure 9.9). Like the Ring Algorithm, the Star Algorithm also converges to a solution, which is almost identical to the actual input spectrum in less than 100 rounds. ◊

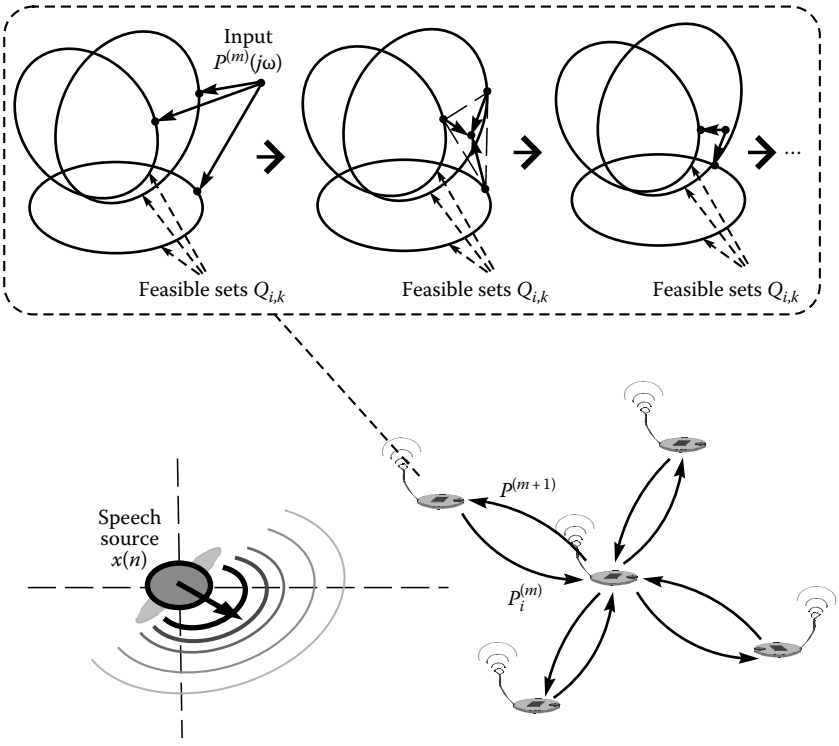


FIGURE 9.8 Star Algorithm. Again, only three feasible sets $Q_{i,k}$ are shown in the inside picture. In practice, each sensor node has to repeat the sequence of projections and averaging many times before it can successfully project the input $P^{(m)}(e^{j\omega})$ supplied by the central node into the intersection of its feasible sets. The projection result, which is called $P_i^{(m)}(e^{j\omega})$ is sent back to the central node. The central node then averages all the $P_i^{(m)}(e^{j\omega})$ it has received and averages them to produce $P^{(m+1)}(e^{j\omega})$. This is sent back to the individual nodes and the process repeats.

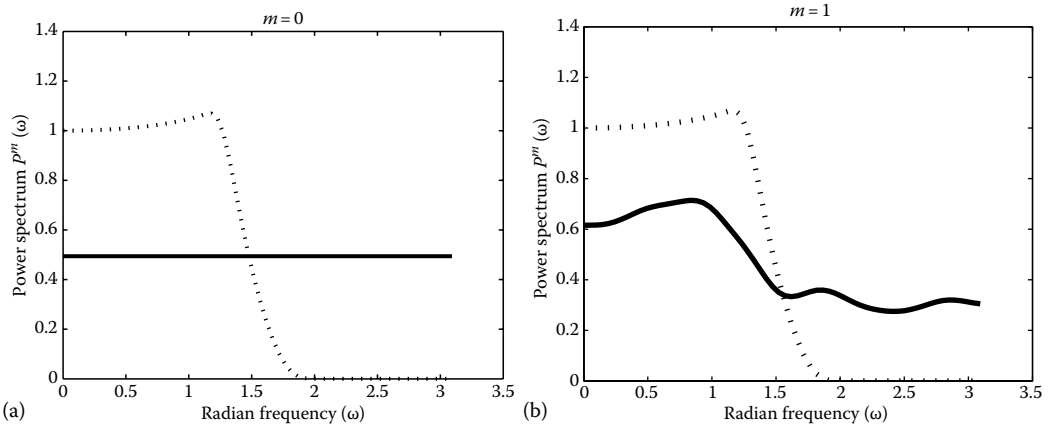


FIGURE 9.9 Star algorithm results.

(*continued*)

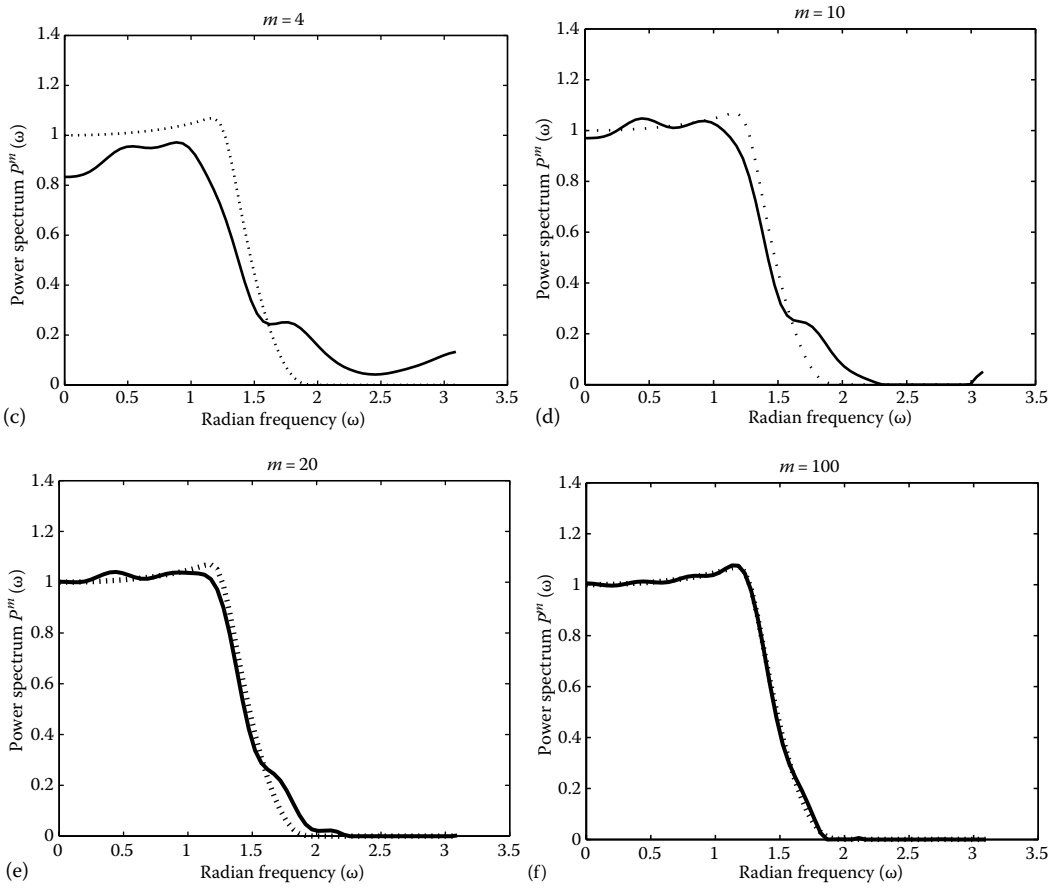


FIGURE 9.9 (continued)

9.6 Conclusion

We introduced a general paradigm for formulating and solving distributed signal-processing problems in sensor network. The core of this paradigm is formulating the problem as a convex feasibility problem. The raw data collected by sensor nodes are processed locally to specify a convex feasible set to which the global solution must belong. Each sensor node in the network specifies its own feasible set and may update this set as it collects new data over time. Information fusion is interpreted as finding a unique and stable global solution in the intersection of the feasibility sets specified by all nodes. The global solution, if it exists, can be found using a plurality of distributed projection algorithms. We discussed two projection algorithms in detail based on the simple RING and STAR network topologies (Figure 9.10).

However, these are not the only possible ways that projection algorithms can be used for solving a convex feasibility problem in a distributed system. It is possible to design many other distributed projection algorithms based on more elaborate network topologies such as those shown in Figure 9.11. Analysis and design of distributed information fusion algorithms for various network topologies are an open field of research. We hope that the initial results presented in our contribution point out the way toward more complete theories and help give shape to the emerging field of sensor processing for sensor networks.

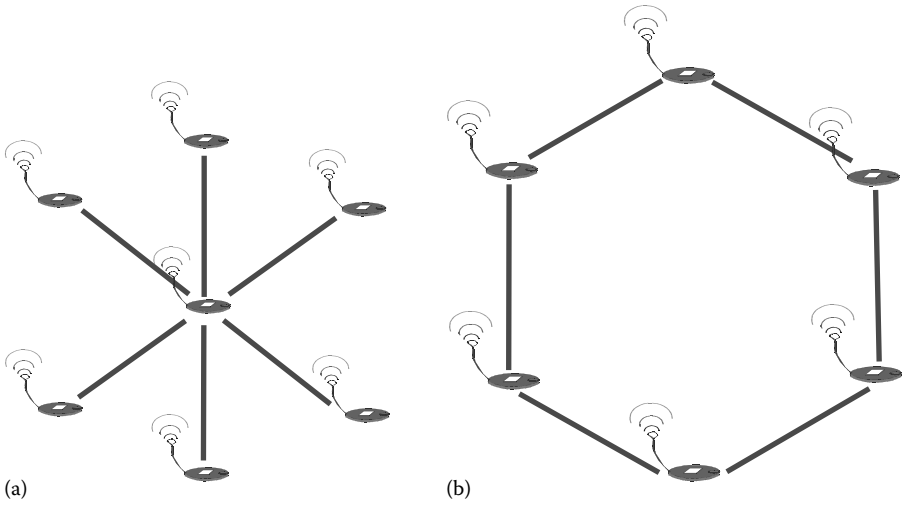


FIGURE 9.10 Two simple network topologies used in this chapter to demonstrate the convex feasibility paradigm: (a) star topology and (b) ring topology.

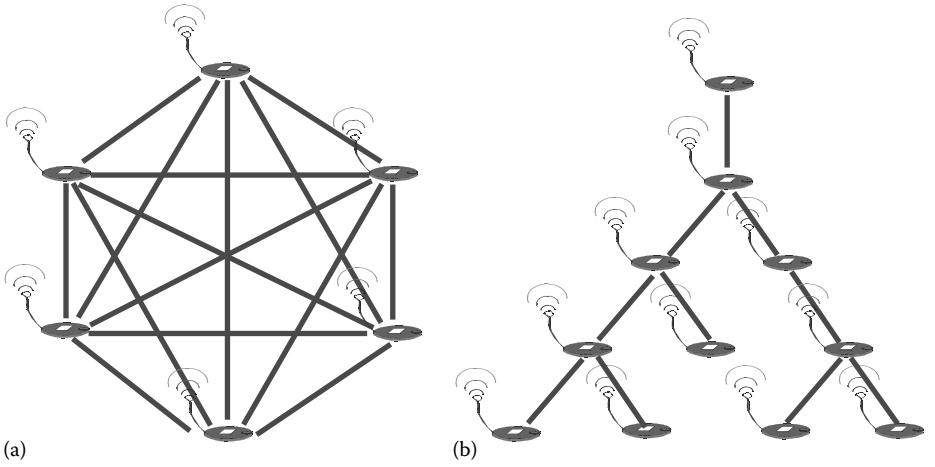


FIGURE 9.11 Other possible topologies for distributed sensor network fusion using the convex feasibility paradigm: (a) fully-connected topology and (b) tree topology.

References

1. S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, U.K., 2004.
2. S. M. Kay, *Modern Spectrum Estimation: Theory and Applications*, Prentice Hall, Upper Saddle River, NJ, 1988.
3. D. B. Percival and A. T. Walden, *Spectral Analysis for Physical Applications*, Cambridge University Press, Cambridge, U.K., 1993.
4. M. H. Hayes, *Statistical Signal Processing and Modeling*, Wiley, New York, 1996.
5. B. Buttkus, *Spectral Analysis and Filter Theory in Applied Geophysics*, Springer-Verlag, Berlin, 2000.

6. O. S. Jahromi, B. A. Francis, and R. H. Kwong, Spectrum estimation using multirate observations, *IEEE Transactions on Signal Processing*, 52(7), 1878–1890, July 2004.
7. A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*, V. H. Winston & Sons, Washington, D.C., 1977.
8. V. V. Vasin and A. L. Ageev, *Ill-Posed Problems with a Priori Information*, VSP, Utrecht, the Netherlands, 1995.
9. H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of Inverse Problems*, Kluwer Academic Publishers, Dordrecht, the Netherlands, 1996.
10. A. N. Tikhonov, A. S. Leonov, and A. G. Yagola, *Nonlinear Ill-Posed Problems*, Chapman and Hall, London, 1998 (2 volumes).
11. K. Chadan, D. Colton, L. Päivärinta, and W. Rundell, *An Introduction to Inverse Scattering and Inverse Spectral Problems*, SIAM, Philadelphia, PA, 1997.
12. V. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1999.
13. F. Jones, *Lebesgue Integration on Euclidean Space*, Jones and Bartlett Publishers, Boston, MA, 1993.
14. J. Hadamard, *Lectures on Cauchy's Problem in Linear Partial Differential Equations*, Yale University Press, New Haven, CT, 1923.
15. A. N. Tikhonov, On solving ill-posed problems and the method of regularization, *Doklady Akademmi Nauk USSR*, 151(3), 501–504, 1963 (in Russian), English translation in *Soviet Math. Dokl.*
16. A. N. Tikhonov, On the regularization of ill-posed problems, *Doklady Akademmi Nauk USSR*, 153(1), 49–52, 1963 (in Russian), English translation in *Soviet Math. Dokl.*
17. D. L. Phillips, A technique for numerical solution of certain integral equations of the first kind, *Journal of the Association for Computing Machinary*, 9, 84–97, 1962.
18. V. K. Ivanov, Integral equations of the first kind and the approximate solution of an inverse potential problem, *Duklady Academmi Nauk. USSR*, 142, 997–1000, 1962 (in Russian), English translation in *Soviet. Math. Dokl.*
19. V. K. Ivanov, On linear ill-posed problems, *Duklady Academmi Nauk. USSR*, 145, 270–272, 1962 (in Russian), English translation in *Soviet. Math. Dokl.*
20. V. V. Vasin, Relationship of several variational methods for approximate solutions of ill-posed problems, *Mathematical Notes*, 7, 161–166, 1970.
21. Y. Censor and S. A. Zenios, *Parallel Optimization: Theory, Algorithms, and Applications*, Oxford University Press, New York, 1997.
22. H. H. Bauschke and J. M. Borwein, On projection algorithms for solving convex feasibility problems, *SIAM Review*, 38, 367–426, 1996.
23. J. M. Borwein and A. S. Lewis, Partially-finite programming in $\mathbf{L}_1$ and the existence of maximum entropy estimates, *SIAM Journal of Optimization*, 3, 248–267, 1993.
24. M. Klaus and R. T. Smith, A Hilbert space approach to maximum entropy regularization, *Mathematical Methods in Applied Sciences*, 10, 397–406, 1988.
25. U. Amato and W. Hughes, Maximum entropy regularization of Fredholm integral equations of the first kind, *Inverse Problems*, 7, 793–808, 1991.
26. J. M. Borwein and A. S. Lewis, Convergence of best maximum entropy estimates, *SIAM Journal of Optimization*, 1, 191–205, 1991.
27. P. P. B. Eggermont, Maximum entropy regularization for Fredholm integral equations of the first kind, *SIAM Journal on Mathematical Analysis*, 24(6), 1557–1576, 1993.
28. M. Teboulle and I. Vajda, Convergence of best ϕ-entropy estimates, *IEEE Transactions on Information Theory*, 39(1), 78–83, 1993.
29. A. S. Leonev, A generalization of the maximal entropy method for solving ill-posed problems, *Siberian Mathematical Journal*, 41(4), 716–724, 2000.
30. N. Wu, *The Maximum Entropy Method*, Springer, Berlin, 1997.

10

Sensor Network Security

10.1 Introduction and Motivation **10**-2
10.2 Denial of Service and Routing Security **10**-4
10.3 Energy Efficient Confidentiality and Integrity **10**-8
10.4 Authenticated Broadcast **10**-12
10.5 Alternative Approaches to Key Management **10**-14
10.6 Secure Data Aggregation **10**-23
10.7 Summary **10**-27
References **10**-28

Guenter Schaefer
Ilmenau University of Technology

This chapter* gives an introduction to the specific security challenges in wireless sensor networks and some of the approaches to overcome them that have been proposed thus far. As this area of research still is a very active one at the time of this writing, it is to be expected that more approaches are going to be proposed as the field gets more mature, so that this chapter should rather be understood as a snapshot than a definitive account of the field.

When thinking of wireless sensor network security, one major question coming into mind is what are the differences between security in sensor networks and "general" network security? In both cases, one usually aims to ensure certain "security objectives" (also called "security goals"). In general, the following objectives are considered to be essential: "authenticity of communicating entities and messages (data integrity), confidentiality, controlled access, availability of communication services, and non-repudiation of communication acts [Sch03]. Basically, these are the same objectives that need to be ensured also in wireless sensor networks (with maybe the exception of non-repudiation which is of less interest at the level on which sensor networks operate). Also, in both cases "cryptographic algorithms and protocols" [MOV97] are the main tool to be deployed for ensuring these objectives. So, from a high level point of view, one could come to the conclusion that sensor network security does not add much to what we already know from network security in general, and thus the same methods could be applied in sensor networks as in classical fixed or wireless networks.

However, closer consideration reveals various differences that have their origins in specific characteristics of wireless sensor networks, so that straightforward application of known techniques is not appropriate. In this chapter we, therefore, first point out these characteristics and give an overview over the specific threats and security challenges in sensor networks. The remaining sections of the chapter then deal in more detail with the identified challenges, that are "denial of service (DoS) and routing security, energy efficient confidentiality and integrity, authenticated broadcast, alternative approaches to key management, and secure data aggregation."

* A prior version of this chapter has been published in *The Embedded Systems Handbook* by CRC Press [Sch05].

10.1 Introduction and Motivation

The main characteristics of wireless sensor networks [KW05] can be explained by summarizing that they are envisaged to be

- Formed by tens to thousands of small, inexpensive sensors that communicate over a wireless interface
- Connected via base stations to traditional networks/hosts running applications interested in the sensor data
- Using multi-hop communications among sensors to bridge the distance between sensors and base stations
- Considerably resource constrained due to limited availability of energy

To get an impression of the processing capabilities of a wireless sensor node, one should have the following example of a sensor node in mind: a node running an 8-bit CPU at 4 MHz clock frequency, 4 kB free of 8 kB flash read only memory, 512 bytes SRAM main memory, a 19.2 kbit/s radio interface and the node being powered by battery.

Typical applications envisaged for wireless sensor networks are environment monitoring (earthquake or fire detection, etc.), home monitoring and convenience applications, site surveillance (intruder detection), logistics and inventory applications (tagging and locating goods, containers, etc.), as well as military applications (battleground reconnaissance, troop coordination, etc.). The fundamental communication pattern to be used in such a network consists of an application demanding some named information in a specific geographical area. Upon this request, one or more base stations broadcast the request, and wireless sensors relay the request and generate answers to it if they contribute to the requested information. The answers are then processed and aggregated as they flow through the network toward the base station(s).

Figure 10.1 shows an exemplary sensor network topology as currently designated for such applications. The sensor network itself consists of one or more base stations that may be able to communicate

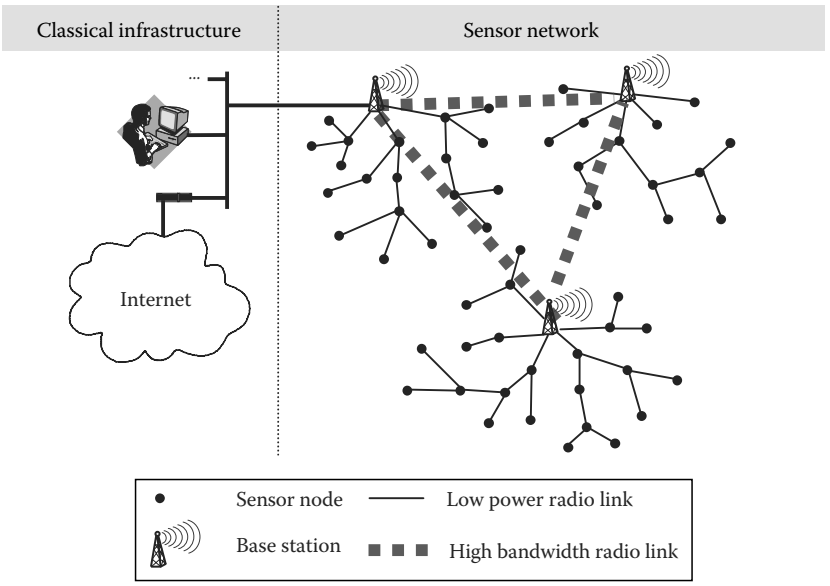


FIGURE 10.1 General sensor network topology example.

among each other by some high bandwidth link (e.g., IEEE 802.11). The base stations furthermore communicate with sensor nodes over a low bandwidth link. As not all sensor nodes can communicate directly with the base station, multi-hop communication is used in the sensor network to relay queries or commands sent by the base station to all sensors, as well as to send back the answers from sensor nodes to the base station. If multiple sensors contribute to one query, partial results may be aggregated as they flow toward the base station. To communicate results or report events to an application residing outside of the sensor network, one or more base stations may be connected to a classical infrastructure network.

As the above description already points out, there are significant differences between wireless sensor and so-called ad hoc networks, to which they are often compared. Both types of networks can be differentiated more specifically by considering the following characteristics [KW03a]:

- Sensor networks show distinctive application-specific characteristics, e.g., depending on its application, a sensor network might be very sparse or dense.
- Interaction of the network with its environment may cause rather bursty traffic patterns. Consider, e.g., a sensor network deployed for detecting/predicting earthquakes or fire detection. Most of the time, there will be little traffic, but if an incident happens the traffic load will increase heavily.
- Scale of sensor networks is expected to vary between tens and thousands of sensors.
- Energy is even more scarce than in ad hoc networks as sensors will be either battery-powered or powered by environmental phenomena (e.g., vibration).
- Self-configurability will be an important feature of sensor networks. While this requirement also exists for ad hoc networks, its importance is even higher in sensor networks, as for example human interaction during configuration might be prohibitive, the geographic position of sensor nodes has to be learned, etc.
- Regarding dependability and quality of service (QoS), classical QoS notions like throughput, jitter, etc. are of little interest in sensor networks, as the main requirement in such networks is the plain delivery of requested information, and most envisaged applications only pose low bandwidth requirements.
- As sensor networks follow a data-centric model, sensor identities are of little interest, and new addressing schemes, e.g., based on semantics or geography, are more appealing.
- Required simplicity of sensor nodes in terms of operating system, networking software, memory footprint, etc. is much more constraining than in ad hoc networks.

Thus far, we have mainly described sensor networks according to their intrinsic characteristics, and regarding their security, we have only stated that principally the same security objectives need to be met as in other types of networks. This leads to the question, what makes security in sensor networks a genuine area of network security research?

To give a short answer, there are three main reasons for this. First, sensor nodes are deployed under particularly harsh conditions from a security point of view, as there will often be a high number of nodes distributed in a (potentially hostile) geographical area, so that it has to be assumed that at least some nodes may get captured and compromised by an attacker. Second, the severe resource constraints of sensor nodes in terms of computation time, memory, and energy consumption demand for very optimized implementation of security services, and also lead to a very unfair power balance between potential attacker (e.g., equipped with a notebook) and defender (cheap sensor node). Third, the specific property of sensor networks to aggregate (partial) answers to a request as the information flows from the sensors toward the base station calls for new approaches for ensuring the authenticity of sensor query results, as established end-to-end security approaches are not appropriate for this.

Consequently, the following security objectives prove to be challenging in wireless sensor networks:

- *Avoiding and coping with sensor node compromise:* this includes measures to partially "hide" the location of sensor nodes at least on the network layer, so that an attacker should ideally not be able to use network layer information to locate specific sensor nodes. Furthermore, sensor nodes should as far as possible be protected from compromise through tamper-proofing measures, where this is economically feasible. Finally, as node compromise cannot be ultimately prevented, other sensor network security mechanisms should degrade gracefully in case of single node compromises.
- *Maintaining availability of sensor network services:* this requires a certain level of robustness against so-called DoS attacks, protection of sensor nodes from malicious energy draining, and ensuring the correct functioning of message routing.
- *Ensuring confidentiality and integrity of data:* data retrieved from sensor networks should be protected from eavesdropping and malicious manipulation. To attain these goals in sensor networks, both efficient cryptographic algorithms and protocols as well as an appropriate key management are required, and furthermore the specific communication pattern of sensor networks (including data aggregation) has to be taken into account.

In the following sections, we will discuss these challenges in more detail and present first approaches that have been proposed to meet them.

10.2 Denial of Service and Routing Security

DoS attacks aim at denying or degrading a legitimate user's access to a service or network resource, or at bringing down the systems offering such services themselves.

From a high level point of view, DoS attacks can be classified into two categories "resource destruction" and "resource exhaustion." In a more detailed examination, the following DoS attacking techniques can be identified:

- Disabling services by
  - Breaking into systems ("hacking")
  - Making use of implementation weaknesses as buffer overrun, etc.
  - Deviation from proper protocol execution
- Resource exhaustion by causing
  - Expensive computations
  - Storage of state information
  - Resource reservations (e.g., bandwidth)
  - High traffic load (requires high overall bandwidth from attacker)

Generally speaking, these attacking techniques can be applied to protocol processing functions at different layers of the protocol architecture of communication systems. While some of the attacking techniques can be defended against by a combination of established means of good system management, software engineering, monitoring, and intrusion detection, the attacking techniques protocol deviation and resource exhaustion require dedicated analysis for specific communication protocols.

In sensor networks, two aspects raise specific DoS concerns: First, breaking into sensor nodes is facilitated by the fact that it might be relatively easy for an attacker to physically capture and manipulate some of the sensor nodes distributed in an area, and second, energy is a very scarce resource in

TABLE 10.1 DoS-Threats in Wireless Sensor Networks [WS02]

Network Layer	Attacks	Countermeasures
Physical	Tampering	Tamper-proofing, hiding
	Jamming	Spread-spectrum, priority messages, lower duty cycle, region mapping, mode change
Link	Collision	Error-correcting code
	Exhaustion	Rate limitation
	Unfairness	Small frames
Network	Neglect and greed	Redundancy, Probing
	Homing	Encryption (only partial protection)
	Misdirection	Egress filtering, authorization, monitoring
	Black holes	Authorization, monitoring, redundancy
Transport	Flooding	Client puzzles
	Desynchronization	Data origin authentication

sensor nodes, so that any opportunity for an attacker to cause a sensor node to wake up and perform some processing functions is a potential DoS vulnerability.

In 2002, Wood and Stankovic published an article on DoS threats in sensor networks [WS02] in which they mainly concentrate on protocol functions of the first four open systems interconnection (OSI) layers. Table 10.1 gives an overview of their findings and potential countermeasures proposed.

On the physical layer, "jamming" of the wireless communication channel represents the principal attacking technique. Spread-spectrum techniques are by nature more resistant against this kind of attack, but can nevertheless not guarantee availability of physical layer services. In case that the bandwidth available in an area is reduced by a DoS attack, giving priority to more important messages could help maintain at least basic operations of a sensor network. While jamming mainly disturbs the availability of sensor nodes to communicate, it has second DoS relevant side effect. As a consequence of worse channel conditions, sensor nodes need more energy to exchange messages. Depending on protocol implementation, this could even lead to energy exhaustion of some nodes, if they tirelessly try to send their messages instead of waiting for better channel conditions. Therefore, from a DoS avoidance point of view, lower duty cycles could be a beneficial protocol reaction to bad channel conditions. Furthermore, the routing protocol (see also below) should avoid to direct messages into jammed areas, and ideally, cooperating sensor nodes located at the edge of a jammed area could collaborate to map jamming reports and reroute traffic around this area. If sensor nodes posses of multiple modes of communication (e.g., wireless and infrared communications), changing the mode is also a potential countermeasure. Finally, even if not directly related to communications, capturing and "tampering" of sensor nodes can also be classified as a physical layer threat. Tamper-proofing of nodes is one obvious measure to avoid further damage resulting from misuse of captured sensor nodes. A traditional preventive measure to at least render capturing of nodes more difficult is to hide them.

On the link layer, Wood and Stankovic identify (malicious) "collisions" and "unfairness" as potential threats and propose as classical measures the use of error-correcting codes and small frames. While one could argue that both threats (and respective countermeasures) are not actually security-specific but also known as conventional problems (and strategies for overcoming them), their deliberate exposure for DoS attacks could nevertheless lead to temporal unavailability of communication services, and ultimately to "exhaustion" of sensor nodes. For the latter threat, the authors propose rate limitation as a potential countermeasure (basically the same idea as lower duty cycle mentioned in the physical layer discussion).

Considering the network layer, threats can be further subdivided into forwarding- and routing-related threats. Regarding forwarding, the main threats are "neglect and greed", that is, sensor nodes that might only be interested in getting their own packets transferred in the network without correctly participating in the forwarding of other node's packets. Such behavior could potentially be detected by the use of probing packets and circumvented by using redundant communication paths. However, both measures increase the network overhead and thus do not come for free. If packets contain

the geographical position of nodes in cleartext, this could be exploited by an attacker for "homing" (locating) specific sensor nodes to physically capture and compromise them. As a countermeasure against this threat, Wood and Stankovic propose encryption of message headers and content between neighboring nodes. Regarding routing-related threats, deliberate "misdirection" of traffic could lead to higher traffic load, as a consequence to higher energy consumption in a sensor network, and potentially also to unreachability of certain network parts. Potential countermeasures against this threat are egress filtering, that is checking the direction in which messages will be routed, authorization verification of routing-related messages, monitoring of routing and forwarding behavior of nodes by neighboring nodes, and redundant routing of messages via multiple paths that in the ideal case do not share common intermediate nodes. The same countermeasures can also be applied to defend against so-called black hole attacks, in which one node or part of the network attracts a high amount of traffic (e.g., by announcing short routes to the base station) but does not forward this traffic.

On the transport layer, the threats "flooding" with connection requests and "desynchronization" of sequence numbers are identified in [WS02]. Both attack techniques are known from classical Internet communications and might potentially also be applied to sensor networks, in case that such networks are going to make use of transport layer connections. Established countermeasures to defend them are so-called client puzzles [TAL01] and authentication of communication partners.

Recapitulating the above discussion, it can be seen that especially the network layer exhibits severe DoS vulnerabilities and proves to be the most interesting layer for potential attackers interested in degrading the availability of sensor network services. This is mostly due to the fact that in this layer the essential forwarding and routing functionality is realized, so that an attacker can cause significant damage with rather moderate means (e.g., in comparison to jamming a large area). In the following, we will therefore further elaborate on this layer and at the same time extend our discussion on general threats on forwarding and routing functions including attacks beyond pure DoS interests.

In Ref. [KW03b], Karlof and Wagner give an overview on attacks and countermeasures regarding secure routing in wireless sensor networks. From a high level point of view, they identify the following threats:

- *Insertion of spoofed, altered or replayed routing information* with the aim of loop construction, attracting, or repelling traffic, etc.
- *Forging of acknowledgments* which may trick other nodes to believe that a link or node is either dead or alive when in fact it is not
- *Selective forwarding* which may be realized either "in-path" or "beneath path" by deliberate jamming, and which allows to control what information is forwarded and what information is suppressed
- *Creation of so-called* "sinkholes," that is attracting traffic to a specific node, e.g., to prepare selective forwarding
- *Simulating multiple identities* ("Sybil attacks"), which allows to reduce effectiveness of fault-tolerant schemes like multipath routing
- *Creation of so-called wormholes* by tunneling messages over alternative low-latency links, e.g., to confuse the routing protocol, create sinkholes, etc.
- *Sending of so-called "hello floods"* (more precisely: "hello shouting"), in which an attacker sends or replays a routing protocol's hello packets with more energy to trick other nodes into the belief that they are neighbors of the sender of the received messages

In order to give an example for such attacks, Figure 10.2 [Woo01] illustrates the construction of a breadth first spanning tree, and Figure 10.3 [KW03b] shows the effect of two attacks on routing schemes that use the breadth first search tree idea to construct their forwarding tables.

One example of a sensor network-operating system that builds a breadth-first spanning tree rooted at the base station is TinyOS. In such networks, an attacker disposing of one or two laptops can either

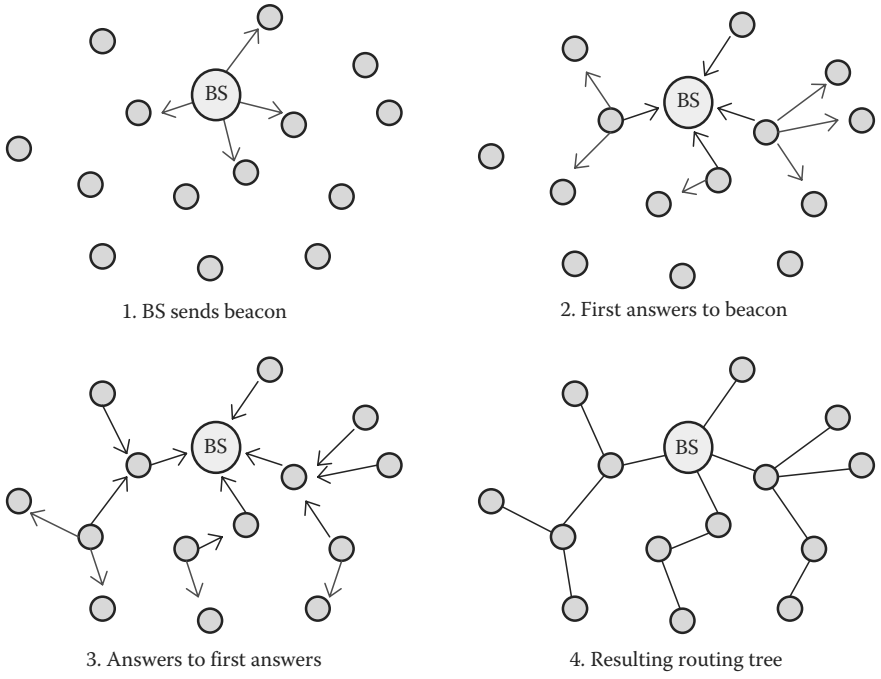


FIGURE 10.2 Breadth first search.

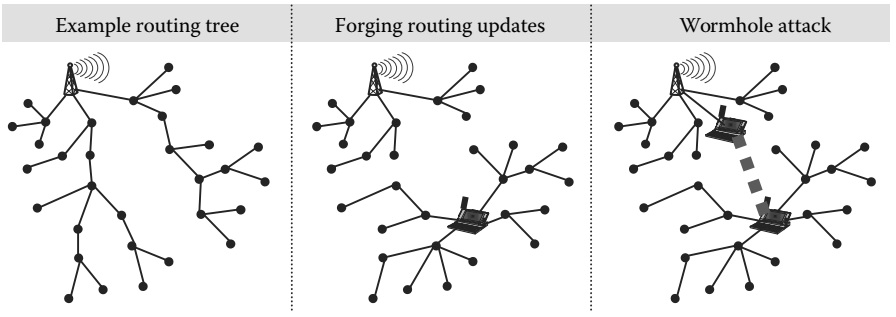


FIGURE 10.3 Attacks on breadth first search routing.

send out forged routing information or launch a wormhole attack. As it can be seen in Figure 10.3, both attacks lead to entirely different routing trees and can be used to prepare further attacks like selective forwarding, etc.

To defend against the abovementioned threats, Karlof and Wagner discuss various methods. Regarding forging of routing information or acknowledgments, data origin authentication and confidentiality of link layer Protocol Data Units (PDUs) can serve as an effective countermeasure. While the first naive approach of using a single group key for this purpose exhibits the rather obvious vulnerability that a single node compromise would result in complete failure of the security, a better, still straightforward approach is to let each node share a secret key with a base station and to have base stations act as trusted third parties in key negotiation (e.g., using the Otway–Rees protocol [OR87]).

Combined with an appropriate key management, the abovementioned link layer security measures could also limit the threat potential of the attack of simulating multiple identities: by limiting

the number of neighbors a node is allowed to have, e.g., through enforcement during key distribution, authentic sensor nodes could be protected from accepting too many neighborhood relations. Additionally, by keeping track of authentic identities and associated keys, the ability of potentially compromised nodes to simulate multiple identities could be restricted. However, the latter idea requires some kind of global knowledge that often can only be realized efficiently by a centralized scheme, which actively involves a base station in the key distribution protocol.

When it comes to hello shouting and wormhole/sinkhole attacks, however, pure link layer security measures cannot provide sufficient protection, as they cannot protect completely against replay attacks. Links should, therefore, be checked in both directions before making routing decisions to defend against simple hello shouting attacks. Detection of wormholes actually proves to be difficult and first approaches to this problem require rather tight clock synchronization [HPJ02]. Sinkholes might be avoided by deploying routing schemes like geographical routing that do not rely on constructing forwarding tables according to distance measured in hops to destination (provided that geographic location coordinates are properly known to sensor nodes). Selective forwarding attacks might be countered with multipath routing. However, this requires redundancy in the network and results in higher network overhead.

In [PLGP06], Parno et al. propose an approach for secure routing in sensor networks that is based on a "recursive grouping algorithm" for address assignment and setup of routing tables. To achieve its security objectives, the approach relies on identity certificates for each sensor node, signed with the private key of a network authority (the corresponding public key is assumed to be known to all sensor nodes), and a hash tree based verification procedure during the recursive grouping phase that assigns the node addresses and builds up the routing tables. Furthermore, when a node detects malicious behavior of another node, it can eliminate the malicious node and itself by sending a signed node revocation message. Please note that in this case the revoking node needs to sacrifice itself and will also be excluded to defend against compromised nodes that revoke multiple sensor nodes. The approach assumes sensor nodes to be able to perform asymmetric cryptographic operations for checking identification certificates, as well as for generating and checking signatures for node revocation messages. The authors propose to make use of asymmetric cryptographic techniques that put most effort on the signer and allow for cheap signature verification, e.g., Rabin signatures [MOV97]. Only if a node revokes another malicious node, it needs to compute a signature itself, requiring significant computational resources and energy. As in such a case, the revoking sensor node itself will also be excluded from the network, the energy drain caused by the signature generation can be tolerated according to the authors of Ref. [PLGP06]. For this idea to work, however, a revoking node needs to be sure, that it is revoking a genuine node of the sensor network with a valid identity and certificate. Otherwise an external attacker aiming to cause DoS could easily provoke as many genuine sensor nodes as he likes to sacrifice themselves with a revoke operation.

10.3 Energy Efficient Confidentiality and Integrity

The preceding discussion of potential countermeasures against DoS attacks and general attacks on routing in wireless sensor networks has shown that the security services confidentiality and integrity prove to be valuable mechanisms against various attacks. Obviously, they are also effective measures to protect application data (e.g., commands and sensor readings) against unauthorized eavesdropping and manipulation, respectively. In this section, we will therefore examine their efficient implementation in resource-restricted sensor networks.

In their paper *SPINS: Security Protocols for Sensor Networks* [PST+02], Perrig et al. discuss the requirements and propose a set of protocols for realizing efficient security services for sensor networks. The main challenges in the design of such protocols arise out of tight implementation constraints in terms of instruction set, memory, CPU speed, a very small energy budget in low-powered

devices, and the fact that some nodes might get compromised. These constraints opt out some well-established alternatives: asymmetric cryptography [DH76,RSA78,ElG85] is generally considered to be too expensive as it results in high computational cost and long ciphertexts and signatures (sending and receiving are very expensive!). Especially, public key management based on certificates exceeds the sensor nodes' energy budget, and key revocation is almost impossible to realize under the restricted conditions in sensor networks. Even symmetric cryptography implementation turns out to be non-straightforward due to architectural limitations and energy constraints. Furthermore, the key management for authenticating broadcast-like communications calls for new approaches, as simple distribution of one symmetric group key among all receivers would not allow to cope with compromised sensor nodes.

Perrig et al. therefore propose two main security protocols:

- *Sensor Network Encryption Protocol (SNEP)* for realizing efficient end-to-end security between nodes and base stations
- Variant of the *Timed Efficient Stream Loss-tolerant Authentication Protocol (TESLA)*, called μ*TESLA*, for authenticating broadcast communications that will be further discussed in Section 10.4

The main goal in the development of SNEP was the efficient realization of end-to-end security services for two-party communication. SNEP provides the security services "data confidentiality," "data origin authentication", and "replay protection." The considered communication patterns are "node to base station" (e.g., sensor readings) and "base station to individual nodes" (e.g., specific requests). Securing messages from a "base station to all nodes" (e.g., routing beacons, queries, reprogramming of the entire network) is the task of the μTESLA protocol to be discussed in Section 10.4. The main design decisions in the development of SNEP were to avoid use of asymmetric cryptography, to construct all cryptographic primitives out of a single block cipher, and to exploit common state to reduce communication overhead where this is possible.

SNEP's basic trust model assumes that two communicating entities A and B share a common master key $X_{A,B}$. Initially, the base station shares a master key with all nodes and node-to-node master keys can be negotiated with the help of the base station (see below). From such a master key, two confidentiality keys $CK_{A,B}, CK_{B,A}$ (one per direction), two integrity keys $IK_{A,B}, IK_{B,A}$, and a random seed $RK_{A,B}$ are derived according to the following equations (for definition of function F see below):

$$\begin{aligned} CK_{A,B} &= F_{X_{A,B}}(1) \\ CK_{B,A} &= F_{X_{A,B}}(3) \\ IK_{A,B} &= F_{X_{A,B}}(2) \\ IK_{B,A} &= F_{X_{A,B}}(4) \\ RK_{A,B} &= F_{X_{A,B}}(5) \end{aligned}$$

The principal cryptographic primitive of SNEP is the RC5 algorithm [BR96]. Three parameters of this algorithm can be configured: the word length $w[bit]$, the number of rounds r, and the key size $b[byte]$, and the resulting instantiation of the algorithm is denoted as *RC5-w/r/b*. What makes RC5 specifically suitable for implementation in sensor nodes is the fact that it can be programmed with a few lines of code and that the main algorithm only makes use of three simple and efficient to execute instructions: two's complement addition + of words $(mod\ 2w)$, bit-wise XOR of words, and cyclic rotation <<<. Figure 10.4 illustrates the encryption function. The corresponding decryption function can be easily obtained by basically "reading the code in reverse." Prior to en- or decryption with RC5, an array $s[0, 2r+1]$ has to be filled by a key preparation routine that is a little bit more tricky, but also uses only simple instructions.

```
// Algorithm:   RC5-Encryption
// Input:       A, B         = plaintext stored in two words
//              S[0, 2r+1]   = an array filled by a key setup
//                             procedure
// Ouput:       A, B         = ciphertext stored in two words

A :=  A + S[0];
B :=  B + S[1];
for i  := 1 to r
     A  := ( (A ⊕ B) <<< B) + s[2i];
     B  := ( (B ⊕ A) <<< A) + s[2i + 1];
```

FIGURE 10.4 RC5 encryption algorithm.

TABLE 10.2 Plaintext Requirements for Differential Attacks on RC5

Number of Rounds	4	6	8	10	12	14	16
Differential attack (chosen plaintext)	2^7	2^{16}	2^{28}	2^{36}	2^{44}	2^{52}	2^{61}
Differential attack (known plaintext)	2^{36}	2^{41}	2^{47}	2^{51}	2^{55}	2^{59}	2^{63}

Regarding the security of the RC5 algorithm, Kaliski and Yin report in 1998 that the best-known attacks against RC5 with a blocklength of 64 bit have plaintext requirements as listed in Table 10.2 [KY98]. According to the information given in [PST+02] (RAM requirements, etc.), Perrig et al. seem to plan for RC5 with 8 rounds and 32 bit words (leading to a blocklength of 64 bit), so that a differential cryptanalysis attack would require about 2^{28} chosen plaintexts and about 2^{47} known plaintexts and CPU effort in the same order of magnitude. Taking into account progress in PC technology, this should be considered on the edge of being secure (if an attacker can collect that many plaintexts). Nevertheless, by increasing the number of rounds the required effort could be raised to 2^{61} or 2^{63}, respectively. Even higher security requirements can by principle only be ensured by using a block cipher with a larger block size.

In SNEP, encryption of messages is performed by using the RC5 algorithm in an operational mode called "counter mode" that XORs the plaintext with a pseudo-random bit sequence, which is generated by encrypting increasing counter values (see also Figure 10.5). The encryption of message *Msg* with key *K* and counter value "*Counter*" is denoted as $\{M\}_{<K,\text{Counter}>}$.

For computing message authentication codes (MACs), SNEP uses the well established *cipher block chaining MAC (CBC-MAC)* construction. This mode encrypts each plaintext block $P_1, \ldots, P_n$ with an integrity key *IK*, XORing the ciphertext of the last encryption result C_{i-1} with the plaintext block P_i prior to the encryption step. The result of the last encryption step is then taken as the MAC (see also Figure 10.6).

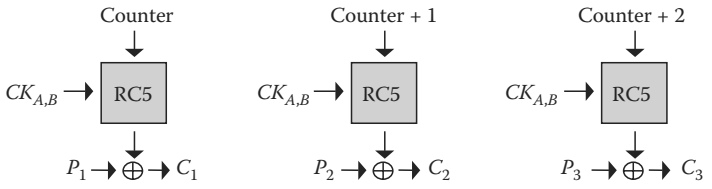


FIGURE 10.5 Encryption in counter mode.

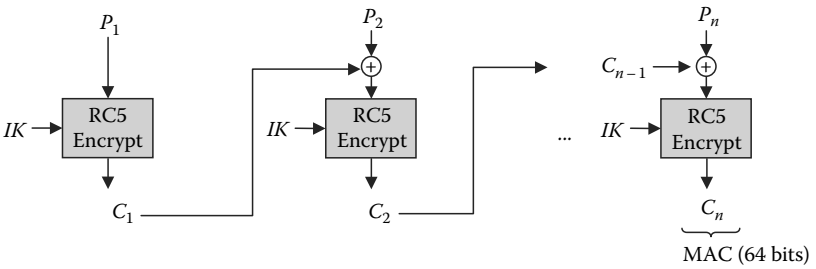


FIGURE 10.6 Computing an MAC in cipher block chaining mode.

Depending on whether encryption of message data is required or not, SNEP offers two message formats:

1. First format appends an RC5-CBC-MAC computed with the integrity key $IK_{A,B}$ over the message data:

$$A \to B: \quad Msg \mid RC5\text{-}CBC(IK_{A,B}, Msg)$$

2. Second format encrypts the message and appends an MAC in whose computation the counter value is also included:

$$A \to B: \quad \{Msg\}_{<CK_{A,B},\text{Counter}>} \mid RC5\text{-}CBC(IK_{A,B}, \text{Counter}, \{Msg\}_{<CK_{A,B},\text{Counter}>})$$

   Please note that the counter value itself is not transmitted in the message, so that common state between sender and receiver is exploited to save transmission energy and bandwidth.

Furthermore, random numbers are generated by encrypting a (different) counter, and the RC5-CBC construction is also used for key derivation, as the key deriving function mentioned above is realized as

$$F_{X_{A,B}}(n) := RC5\text{-}CBC(X_{A,B}, n)$$

To be able to successfully decrypt a message, the receiver's decryption counter needs to be synchronized with the sender's encryption counter. An initial counter synchronization can be achieved by the following protocol, in which both entities A and B communicate their individual counter value for encryption C_A and C_B to the other party, and authenticate both values by exchanging two MACs computed with their integrity keys $IK_{A,B}$ and $IK_{A,B}$, respectively:

$$\begin{aligned} A \to B: &\quad C_A \\ B \to A: &\quad C_B \mid RC5\text{-}CBC(IK_{B,A}, C_A, C_B) \\ A \to B: &\quad RC5\text{-}CBC(IK_{A,B}, C_A, C_B) \end{aligned}$$

In case of a message loss, counters get out of synch. By trying out a couple of different counter values, a few message losses can be tolerated. However, as this consumes energy, after trying out a couple of succeeding values, an explicit resynchronization dialog is initiated by the receiver A of a message. The dialog consists of sending a freshly generated random number N_A to B, who answers with his current counter C_B and an MAC computed with his integrity key over both the random number and the counter value:

$$\begin{aligned} A \to B: &\quad N_A \\ B \to A: &\quad C_B \mid RC5\text{-}CBC(IK_{B,A}, N_A, C_B) \end{aligned}$$

As encrypted messages are only accepted by a receiver if the counter value used in their MAC computation is higher than the last accepted value, the implementation of the confidentiality service in SNEP to a certain degree also provides replay protection. If for a specific request Req an even tighter time synchronization is needed, the request can also contain a freshly generated random number N_A that will be included in the computation of the MAC of the answer message containing the response Rsp:

$$\begin{aligned} A \rightarrow B: \quad & N_A, Req \\ B \rightarrow A: \quad & \{Rsp\}_{<CK_{B,A},C_B>} \mid \\ & RC5\text{-}CBC(IK_{B,A}, N_A, C_B, \{Rsp\}_{<CK_{B,A},C_B>}) \end{aligned}$$

To establish a shared secret $SK_{A,B}$ between two sensor nodes A and B with the help of base station BS, SNEP provides the following protocol:

$$\begin{aligned} A \rightarrow B: \quad & N_A \mid A \\ B \rightarrow BS: \quad & N_A \mid N_B \mid A \mid B \mid \\ & RC5\text{-}CBC(IK_{B,BS} \mid N_A \mid N_B \mid A \mid B) \\ BS \rightarrow A: \quad & \{SK_{A,B}\}_{K_{BS,A}} \mid \\ & RC5\text{-}CBC(IK_{BS,A} \mid N_A \mid B \mid \{SK_{A,B}\}_{K_{BS,A}}) \\ BS \rightarrow B: \quad & \{SK_{A,B}\}_{K_{BS,B}} \mid \\ & RC5\text{-}CBC(IK_{BS,B} \mid N_B \mid A \mid \{SK_{A,B}\}_{K_{BS,B}}) \end{aligned}$$

In this protocol, A first sends a random number N_A and his name to B, who in turn sends both values together with his own random number N_B and name B to the base station. The base station then generates a session key $SK_{A,B}$ and sends it to both sensor nodes in two separate messages, which are encrypted with the respective key the base station shares with each node. The random numbers N_A and N_B allow both sensor nodes to verify the freshness of the returned message and the key contained in it.

Regarding the security properties of this protocol, however, it has to be remarked that in a strict sense the protocol as formulated in [PST+02] does neither allow A nor B to perform concurrent key negotiations with multiple entities, as in such a case they would not be able to securely relate the answers to the correct protocol run (please note that the name of the peer entity is not transmitted in the returned message but only included in the MAC computation). Furthermore, neither A nor B knows, if the other party received the key and trusts in its suitability, which is commonly regarded as an important objective of a key management protocol [GNY90]. Finally, the base station cannot deduce anything about the freshness of messages and can therefore not differentiate between fresh and replayed requests for a session key.

10.4 Authenticated Broadcast

Authenticated broadcast is required if one message needs to be sent to all (or many) nodes in a sensor network, and the sensor nodes have to be able to verify the authenticity of the message. Examples for this communication pattern are authenticated query messages, routing beacon messages, or commands to reprogram an entire network. Because it has to be ensured that recipients of such a message should not be able to make use of their verifying key for forging authenticated messages, an asymmetric mechanism has to be deployed. As pointed out before, classical asymmetric cryptography, however, is considered to be too expensive in terms of computation, storage, and communication requirements for sensor nodes.

One basic idea for obtaining asymmetry, while at same time deploying a symmetrical cryptographic algorithm, is to send a message that has been authenticated with a key K_i and to disclose this key at a later point in time, so that the authenticity of the message can be verified. Of course, from the moment on in which the key disclosure message has been sent, a potential attacker could use this key to create MACs for forged messages. Therefore, it is important that all receivers have at least loosely synchronized clocks and only use a key K_i to verify messages that have been received before the key disclosure message was sent.

However, it must also be ensured that a potential attacker cannot succeed in tricking genuine nodes into accepting bogus authentication keys generated by himself. One elegant way to achieve this is the inverse use of a "chain of hash codes" for obtaining integrity keys, basically a variation of the so-called "one-time password" idea [HMNS98].

The *TESLA* protocol uses a reversed chain of hash values to authenticate broadcast data streams [PT03]. The μTESLA protocol proposed to be used in sensor networks is a minor variation of the TESLA protocol, with the basic difference being the cryptographic scheme used to authenticate the initial key. While TESLA uses asymmetric cryptography for this, μTESLA deploys the SNEP protocol, so that the base station calculates for each sensor node one individual MAC that authenticates the initial key K_0. Furthermore, while TESLA discloses the key in every packet, μTESLA discloses the key only once per time interval to reduce protocol overhead, and only base stations authenticate broadcast packets because sensor nodes are not capable of storing entire key chains (see also below).

To setup a sender, first the length n of the key chain to be computed is chosen and the last key of the key chain K_n is randomly generated. Second, the entire hash key chain is computed according to the equation $K_{n-1} := H(K_n)$, stored at the sender, and the key K_0 is communicated and authenticated to all participating sensor nodes. For this, each sensor node A sends a random number N_A to the base station and the base station answers with a message containing its current time, the currently disclosed key K_i (in the initial case: $i = 0$), the time period T_i in which K_i was valid for authenticating messages, the interval length T_{Int}, the number of intervals δ the base station waits before disclosing a key, and an MAC computed with the integrity key $K_{BS,A}$ over these values:

$$\begin{aligned} A \rightarrow BS: &\quad N_A \mid A \\ BS \rightarrow A: &\quad T_{BS} \mid K_i \mid T_i \mid T_{\text{Int}} \mid \delta \mid \\ &\quad RC5\text{-}CBC(IK_{BS,A} \mid N_A \mid T_{BS} \mid K_i \mid T_i \mid T_{\text{Int}} \mid \delta) \end{aligned}$$

After this preparatory phase, broadcasting authenticated packets is then realized as follows:

- Time is divided in uniform length intervals T_i and all sensor nodes are loosely synchronized to the clock of the base station.
- In time interval T_i, the sender authenticates packets with key K_i.
- Key K_i is disclosed in time interval $i + \delta$ (e.g., $\delta = 2$).

Figure 10.7 illustrates this reverse use of the chain of hash values for authenticating packets. To check the authenticity of a received packet, a sensor node first has to store the packet together with T_i and wait until the respective key has been disclosed by the base station. Upon disclosure of the appropriate key K_i, the authenticity of the packet can be checked.

Of course, it is crucial to discard all packets that have been authenticated with an already disclosed key for this scheme to be secure. This requires at least a loose time synchronization with an appropriate value of δ that needs to be selecting in accordance with the maximum clock drift. However, as nodes cannot store many packets, key disclosure cannot be postponed for a long time so that the maximum clock drift should not be too big.

If a sensor node should need to send a broadcast packet, it would send a SNEP-protected packet to the base station, which in turn would then send an authenticated broadcast packet. The main reason

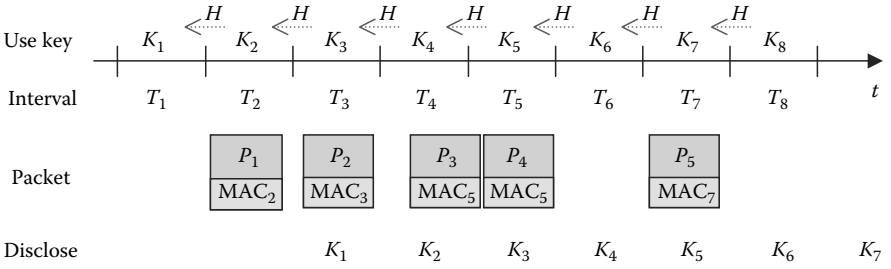


FIGURE 10.7 Example of TESLA operation.

for this is that sensor nodes do not have enough memory for storing key chains and cannot, therefore, authenticate broadcast packets on their own.

In [LN04], Liu and Ning extend the μTESLA idea by proposing "multilevel key chains" to overcome the scalability problems caused by the tradeoff between key chain length and duration of the key disclosure time interval: Shorter duration of the time interval on the one hand has the advantage of sensor nodes being able to check the authenticity of messages and deliver them earlier, thus requiring less buffer space in sensor nodes, but requires rather long key chains to support a given network lifetime. On the other hand, if the interval length is set to a higher value, a shorter key chain can be used, requiring less computation and storage effort for setup of the scheme, but sensor nodes need to buffer more messages before authenticity can be checked. To overcome the problem of long key chains, Liu and Ning propose to use higher-level key chains with long intervals for authenticating commitments of lower-level key chains that are communicated to the sensor nodes early enough to be authenticated before the time of their usage.

10.5 Alternative Approaches to Key Management

Key management is often said to be the hardest part of implementing secure communications, as on the one hand legitimate entities need to hold or be able to agree on the required keys, and on the other hand, a suite of security protocols cannot offer any protection if the keys fall in the hands of an attacker. The SNEP protocol suite as described in Section 10.3 includes a simple and rather traditional key management protocol that enables two sensor nodes to obtain a shared secret key with the help of a base station. In this section, we will treat the subject of key management in more depth and review alternative approaches to it.

All in all, key management comprises of the following tasks [Sch03]:

- "Key generation" is the creation of the keys that are used. This process must be executed in a "random" or at least "pseudo-random-controlled" way, because hackers will otherwise be able to execute the process themselves and in a relatively short time will discover the key that was used for security. Pseudo-random-controlled key generation means that keys are created according to a deterministic approach but each possible key has the same probability of being created from the method. Pseudo-random generators must be initialized with a real random value so that they do not always produce the same keys. If the process of key generation is not reproducible, it is referred to as "really random" key generation.
- Task of "key distribution" consists of deploying generated keys in the place in a system where they are needed. In simple scenarios, the keys can be distributed through direct (e.g., personal) contact. If larger distances are involved and symmetric encryption

algorithms are used, the communication channel again has to be protected through encryption. Therefore, a key is needed for distributing keys. This necessity supports the introduction of what is called "key hierarchies".

- When implementing "key storage", measures are needed to make sure that they cannot be read by unauthorized users. One way to address this requirement is to ensure that the key is regenerated from an easy to remember but sufficiently long password (usually an entire sentence) before each use, and therefore is only stored in the memory of the respective user. Another possibility for storage is manipulation-safe crypto-modules that are available on the market in the form of processor chip cards at a reasonable price.
- "Key recovery" is the reconstruction of keys that have been lost. The simplest approach is to keep a copy of all keys in a secured place. However, this creates a possible security problem because an absolute guarantee is needed that the copies of the keys will not be tampered with. The alternative is to distribute the storage of the copies to different locations, which minimizes the risk of fraudulent use so long as there is an assurance that all parts of the copies are required to reconstruct the keys.
- "Key invalidation" is an important task of key management, particularly with asymmetric cryptographic methods. If a private key is known, then the corresponding public key needs to be identified as invalid. In sensor networks, key invalidation is expected to be a quite likely operation, as sensor nodes may be relatively easy to capture and compromise.
- "Destruction of no longer required keys" is aimed at ensuring that messages ciphered with them also cannot be decrypted by unauthorized persons in the future. It is important to make sure that all copies of the keys have really been destroyed. In modern operating systems, this is not a trivial task as storage content is regularly transferred to hard disk through automatic storage management and the deletion in memory gives no assurance that copies of the keys no longer exist. In the case of magnetic disk storage devices and so-called EEPROMs (Electrically Erasable Programmable Read-Only Memory), these have to be overwritten or destroyed more than once to guarantee that the keys stored on them can no longer be read, even with sophisticated technical schemes.

From the above listed tasks, most key management protocols address the task of key distribution and sometimes also concern key generation. Approaches to distributing keys in traditional networks, however, do not work well in wireless sensor networks. Methods based on asymmetric cryptography require very resource intensive computations and are, therefore, often judged as being not appropriate for sensor networks. Arbitrated key management based on predetermined keys, like the key management protocol of SNEP discussed above, on the other hand, assume predetermined keys at least between the base station and sensor nodes. This requires pre-distribution of these keys before deployment of the sensor network and also has some security implications in case of node compromise.

There are a couple of particular requirements to key management schemes for sensor networks resulting from their specific characteristics [CPS03]:

- *Vulnerability of nodes to physical capture and node compromise:* Sensor nodes may be deployed in difficult to protect/hostile environments and can therefore fall into the hands of an attacker. Because of tight cost constraints, nodes will often not be tamper-proof, so that cryptographic keys might be captured by an attacker. This leads to the requirement that compromises some nodes and keys should not compromise the overall network's security ("graceful degradation").
- *Lack of a-priori knowledge of deployment configuration:* In some applications, sensor networks will be installed via random scattering (e.g., from an airplane), so that neighborhood relations are not known a priori. Even with manual installation, pre-configuration

of sensor nodes would be expensive in large networks. This leads to the requirement that sensor networks' key management should support for "automatic" configuration after installation.

- *Resource restrictions:* As mentioned before, nodes of a sensor network only possess of limited memory and computing resources, as well as very limited bandwidth and transmission power. This puts tight constraints on the design of key management procedures.
- *In-network processing:* Over-reliance on a base station as source of trust may result in inefficient communication patterns (cf. data aggregation in Section 10.6). Also, it turns base stations into attractive targets (which they are in any case!). Therefore, centralistic approaches like the key management protocol of SNEP should be avoided.
- *Need for later addition of sensor nodes:* Compromise, energy exhaustion, or limited material/calibration lifetime may make it necessary to add new sensors to an existing network. However, legitimate nodes that have been added to a sensor network should be able to establish secure relationships with existing nodes. Erasure of master keys after initial installation (cf. the *Localized Encryption and Authentication Protocol* (LEAP) approach described below) does not allow this.

In the following, we will describe two new alternatives to traditional key management approaches that have been proposed for sensor networks: the neighborhood-based initial key exchange protocol *LEAP*, and the approach of "probabilistic key distribution" together with its numerous variations.

The LEAP [ZSJ03] enables "automatic" and efficient establishment of security relationships in an initialization phase after installation of the nodes. It supports key establishment for various trust relationships between:

- Base station and sensor with so-called individual keys
- Sensors that are direct neighbors with "pairwise-shared keys"
- Sensors that form a cluster with "cluster keys"
- All sensors of a network with a "group key"

To establish individual keys prior to deployment, every sensor node u is preloaded with an individual key K_u^m known only to the node and the base station. The base station s generates these keys from a master key K_s^m and the node identity u according to the equation $K_u^m := f(K_s^m, u)$. Generating all node keys from one master key is supposed to save memory at the base station, as the individual keys need not be stored at the base station but can be generated on-the-fly when they are needed.

In scenarios in which pairwise-shared keys cannot be preloaded into sensor nodes because of installation by random scattering but neighboring relationships remain static after installation, LEAP provides for a simple key establishment procedure for neighboring nodes. For this, it is assumed that there is a minimum time interval $T_{\min}$ during which a node can resist against attacks. After being scattered in the field, sensor nodes establish neighboring relations during this time interval based on an initial group key K_I that has been pre-configured into all sensor nodes before deployment. First, every node u computes its master key $K_u = f(K_I, u)$. Then, every node discovers its neighbors by sending a message with his identity u and a random number r_u and collecting the answers:

$$u \to * : \quad u \mid r_u$$
$$v \to u : \quad v \mid MAC(K_v, r_u \mid v)$$

As u can also compute K_v, it can directly check this MAC and both nodes compute the common-shared secret $K_{u,v} := f(K_v, u)$. After expiration of the timer $T_{\min}$, all nodes erase the initial group key K_I and all computed master keys so that only the pairwise-shared keys are kept. This scheme can be extended with all nodes forwarding also the identities of their neighbors, enabling a node also to compute pairwise-shared keys with nodes that are one hop away.

To establish a cluster key with all its immediate neighbors, a node randomly generates a cluster key K_u^c and sends it individually encrypted to all neighbors $v_1, v_2, \ldots$:

$$u \to v_i : \quad E(K_{u,v_i}, K_u^c)$$

All nodes v_i decrypt this message with their pairwise-shared key K_{u,v_i} and store the obtained cluster key. When a node is revoked, a new cluster key is distributed to all remaining nodes.

If a node u wants to establish a pairwise-shared key with a node c that is multiple hops away, it can do so by using other nodes known to it as proxies. To detect suitable proxy nodes v_i, u broadcasts a query message with its own node id and the node id of c. Nodes v_i knowing both nodes u and c will answer to this message:

$$\begin{aligned} u &\to * : \quad u \mid c \\ v_i &\to u : \quad v_i \end{aligned}$$

Assuming that node u has received m answers, it then generates m shares $sk_1, \ldots, sk_m$ of the secret key $K_{u,c}$ to be established with c and sends them individually over the appropriate nodes v_i:

$$\begin{aligned} u &\to v_i : \quad E(K_{u,v_i}, sk_i) \mid f(sk_i, 0) \\ v_i &\to c : \quad E(K_{v_i,c}, sk_i) \mid f(sk_i, 0) \end{aligned}$$

The value $f(sk_i, 0)$ allows the nodes v_i and c to verify if the creator of such a message actually knew the key share sk_i, as otherwise it would not have been able to compute this value (the function f needs to be a one-way function for this to be secure). After receiving all values sk_i, node c computes $K_{u,c} := sk_1 \oplus \cdots \oplus sk_m$.

To establish a new group key K_g, the base station s randomly generates a new key and sends it encrypted with its own cluster key to its neighbors:

$$s \to v_i : \quad E(K_s^c, K_g)$$

All nodes receiving such a message forward the new group key encrypted with their own cluster key to their neighbors.

Node revocation is performed by the base station and uses μTESLA. All nodes, therefore, have to be preloaded with an authentic initial key K_0, and loose time synchronization is needed in the sensor network. To revoke a node u, the base station s broadcasts the following message in time interval T_i using the μTESLA key K_i valid for that interval:

$$s \to * : \quad u \mid f(K'_g, 0) \mid MAC(K_i, u \mid f(K'_g, 0))$$

The value $f(K'_g, 0)$ later on allows all nodes to verify the authenticity of a newly distributed group key K'_g. This revocation becomes valid after disclosure of the TESLA key K_i.

A couple of remarks to some security aspects of LEAP have to be mentioned at this point:

- As every node u knowing K_I may compute the master key K_v of every other node v, there is little additional security to be expected from distinguishing between these different "master keys." Especially, all nodes need to hold K_I during the discovery phase to be able to compute the master keys of answering nodes. The authors of Ref. [ZSJ03] give no reasoning for why they think that this differentiation of master keys should attain any additional security. As any MAC construction should not leak information about K_I in a message authentication code $MAC(K_I, r_u \mid v)$, it is hard to see any benefit in this (is it "crypto snake oil"?).

- The synchronization of the time interval for pairwise key negotiation is critical. However, the authors of Ref. [ZSJ03] give no hint on how the nodes should know when this time interval starts, or if there should be a signal and if so, what to do if a node misses this signal or "sleeps" during the interval? It is clear that if any node is compromised before erasure of K_I the approach fails to provide protection against disclosure of pairwise-shared keys.
- It does not become clear, what is the purpose of the random value (nonce) in the pairwise-shared key establishment dialog. Pairwise-shared keys are only established during $T_{\min}$, and most probably, all neighbors will answer to the first message anyway (including the same nonce from this message). This random value is not even included in the computation of $K_{u,v}$, and so the only thing that can be defended against with it is an attacker that sends replayed replies during $T_{\min}$, but these would not result in additional storage of keys $K_{u,v}$ or anything else than having to parse and discard these replays.
- The cluster key establishment protocol does not allow a node to check the authenticity of the received key, as every attacker could send some binary data that are decrypted to "something." This would overwrite an existing cluster key K_u^c with garbage, leading to a DoS vulnerability. By appending an MAC, this could be avoided. However, an additional replay protection would be required in this case to avoid overwriting with old keys.

Furthermore, after expiration of the initial time intervall $T_{\min}$, it is no longer possible to establish pairwise-shared keys among neighbors, so that the LEAP approach does not support later addition/exchange of sensor nodes.

In 2002, Eschenauer and Gligor proposed a "probabilistic key management scheme" [EG02] that is based on the simple observation that on the one hand, sharing one key K_G among all sensors leads to weak security, and on the other hand, sharing individual keys $K_{i,j}$ among all nodes i, j requires too many keys in large sensor networks ($n^2 - n$ keys for n nodes). The basic idea of probabilistic key management is to randomly give each node a so-called key ring containing a relatively small number of keys from a large key pool, and to let neighboring nodes discover the keys they share with each other. By properly adjusting the size of the key pool and the key rings, a "sufficient" degree of "shared key connectivity" for a given network size can be attained.

The basic scheme published in Ref. [EG02] consists of three phases:

- Key predistribution
- Shared-key discovery
- Path key establishment

The key pre-distribution consists of five steps that are processed offline. First, a large key pool P with about 2^{17} to 2^{20} keys and accompanying key identifiers is generated. Then, for each sensor k keys are randomly selected out of P without replacement, to establish the sensor's key ring. Every sensor is loaded with its key ring comprising the selected keys and their identifiers. Furthermore, all sensor identifiers and the key identifiers of their key ring are loaded into a controller node. Finally, a shared key for secured communication with each sensor s is loaded into the controller node ci, according to the following rule: If $K_1, \ldots, K_k$ denote the keys on the key ring of sensor s, the shared key $K_{ci,s}$ is computed as $K_{ci,s} := E(K_1 \oplus \cdots \oplus K_k, ci)$.

The main purpose of the key predistribution is to enable any two sensor nodes to identify a common key with a certain probability. This probability, that two key rings $KR1$, $KR2$ share at least one common key, can be computed as follows:

$$\Pr(KR1 \;\&\; KR2 \text{ share at least one key}) = 1 - \Pr(KR1 \;\&\; KR2 \text{ share no key})$$

The number of possible key rings is

$$\binom{P}{k} = \frac{P!}{k!(P-k)!}$$

The number of possible key rings after k keys have been drawn from the key pool without replacement is

$$\binom{P-k}{k} = \frac{(P-k)!}{k!(P-2k)!}$$

Thus, the probability that no key is shared is the ratio of the number of key rings without a match divided by the total number of key rings. Concluding the probability of at least one common key is

$$\Pr(\text{at least one common key}) = 1 - \frac{k!(P-k)!(P-k)!}{P!\ k!(P-2k)!}$$

After being installed, all sensor nodes start discovering their neighbors within the wireless communication range, and any two nodes, wishing to find out if they share a key, simply exchange lists of key ids on their key ring. Alternatively, each node could broadcast a list:

$$s \to *: \quad \alpha \mid E(K_1, \alpha) \mid \cdots \mid E(K_k, \alpha)$$

A node receiving such a list would then have to try all its keys, to find out matching keys (with a high probability). This would hide from an attacker which node holds which key ids but requires more computational overhead from each sensor node. The shared key discovery establishes a (random graph) topology in which links exist between nodes that share at least one key. It might happen that one key is used by more than one pair of nodes.

In the path key establishment phase, path keys are assigned to pairs of nodes (s_1, s_n) that do not share a key but are connected by two or more links so that there is a sequence of nodes which share keys and "connect" s_1 to s_n. The article [EG02], however, does not contain any clear information on how path keys are computed or distributed. It only states that they do not need to be generated by the sensor nodes. Furthermore, it is mentioned that "the design of the DSN ensures that, after the shared key discovery phase is finished, a number of keys on any ring are left unassigned to any link." However, it does not become clear from [EG02] how two nodes can make use of these unused keys for establishing a path key.

If a node is detected to be compromised, all keys on its ring need to be revoked. For this, the controller node generates a signature key K_e and sends it individually to every sensor node si, encrypted with the key $K_{ci,si}$:

$$ci \to si: \quad E(K_{ci,si}, K_e)$$

Afterwards, it broadcasts a signed list of all identifiers of keys that have to be revoked:

$$ci \to *: \quad id_1 \mid id_2 \mid \cdots \mid id_k \mid MAC(K_e, id_1 \mid id_2 \mid \cdots \mid id_k)$$

Every node receiving this list has to delete all listed keys from his key ring. This removes all links to the compromised node plus some more links from the random graph. Every node that had to remove some of its links afterwards tries to reestablish as much as possible of them by starting a shared key discovery and a path key establishment phase.

In Ref. [CPS03], Chan et al. propose a modification to the basic random pre-distribution scheme described thus far by requiring to combine multiple-shared keys. In this variant, two nodes are

required to share at least q keys on their rings, to establish a link. So, if $K_1, ..., K_{q'}$ are the common keys of nodes u and v (with $q' \geq q$), then the link key is computed as follows:

$$K_{u,v} := h(K_1, ..., K_{q'})$$

On the one hand, it becomes harder with this scheme for an attacker to make use of one or multiple key ring(s) obtained by node compromise, and this increase in difficulty is exponential in q. On the other hand, the size of the key pool $|P|$ has to be decreased to have a high enough probability that two nodes share enough keys on their rings to establish a link. This gives an attacker a higher percentage of compromised keys per compromised nodes. In Ref. [CPS03], a formula is derived how to compute the key pool size so that any two nodes share enough keys with a given probability $> p$. This scheme is called the "*q-composite scheme*."

In the same paper, Chan et al. propose a second scheme, called "multipath key reinforcement." The basic idea of this scheme is to "strengthen" an already established key by combining it with random numbers that are exchanged over alternative secure links. After the discovery phase of the basic scheme has been completed and enough routing information can be exchanged so that a node u knows all (or at least enough) disjoint paths $p_1, ..., p_j$ to a node v, node u generates j random values $v_1, ..., v_j$ and sends each value along another path to node v. After having received all j values, node v computes the new link key:

$$K'_{u,v} = K_{u,v} \oplus v_1 \oplus \cdots \oplus v_j$$

Clearly, the more paths are used, the harder it gets for an attacker to eavesdrop on all of them. However, the probability for an attacker to be able to eavesdrop on a path increases with the length of the path, so that utilizing more but longer paths does not necessarily increase the overall security to be attained by the scheme. In Ref. [CPS03], the special case of 2-hop multipath key reinforcement is analyzed using probability theory. Furthermore, the paper also describes a third approach called "random pairwise key scheme" that hands out keys to pairs of nodes which also store the identity of the respective peer node holding the same key. The motivation behind this approach is to allow for node-to-node authentication (see Ref. [CPS03] for details).

Concerning security, the following remarks on probabilistic key management should be noted. The nice property of having a rather high probability that any two given nodes share at least one key (e.g., $p = 0.5$, if 75 keys out of 10,000 keys are given to every node) also plays in the hands of an attacker who compromises a node, and an attacker that has compromised more than one node has an even higher probability of holding at least one key with any given node. It has been shown [CPS03] that on the average kn/P nodes share a common key if for each of n nodes k keys are chosen from a key pool of size P. This problem also exists with the q-composite scheme, as the key pool size is reduced to ensure a high enough probability that any two nodes share at least q keys. This especially concerns the attacker's ability to perform active attacks, as eavesdropping attacks are less probable because the probability that the attacker holds exactly the key that two other nodes are using is smaller (and even smaller in the q-composite scheme).

To avoid that attackers can use captured keys to eavesdrop on communication of nodes that have not yet been compromised, Huang and Du developed a node-based probabilistic key distribution scheme that ensures that different keys are used on different links [HD05]. The main idea of this scheme is instead of selecting keys from a large key pool for each node, the set of keys to be given to each node is computed by first randomly choosing a set of t potential peers for each node and then assigning randomly generated keys for each pair of peers. This construction ensures that no key may be used on two different links, so that keys captured from compromised nodes cannot be exploited for eavesdroping on communications of other nodes.

A different construction that aims to ensure the same property is based on "Blom's matrix-based key pre-distribution scheme" [Blo85]. The scheme guarantees the security property that as long as

no more than λ nodes are compromised, the information obtained from these nodes cannot be used to compute keys to be used between pairs of uncompromised nodes. The scheme first constructs a $(\lambda + 1) \times N$ matrix G over a finite field $GF(q)$, where N is the size of the network. The matrix G is assumed to be known to all nodes (actually, each node needs to know one column of G). To save memory, the matrix can be constructed in a special way so that nodes can compute it from a small set of parameters and do not need to store it [MS77]. Let q be the smallest prime that is larger than $2^{\text{keylength}}$. The construction supports networks up to size q, as it has to be ensured that $N < q$. Let s be a "generator" of the group $GF(q)$ that is by computing s^i for all i between 1 and $|GF(q)|$ the whole group $GF(q)$ can be enumerated. The matrix G can be computed according to the following equation:

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 \\ s & s^2 & s^3 & s^N \\ s^2 & (s^2)^2 & (s^3)^2 & (s^N)^2 \\ \cdots & \cdots & \cdots & \cdots \\ s^\lambda & (s^2)^\lambda & (s^3)^\lambda & (s^N)^\lambda \end{pmatrix}$$

As s is a generator of $GF(\mathrm{q})$, it holds that $i \neq j \implies s^i \neq s^j$. G is thus a Vandermonde matrix with $s, s^2, \ldots, s^N$ being all distinct. It has been shown that under these conditions the matrix has maximum rank and any $\lambda + 1$ columns are linearly independent. For the purposes of Blom's key predistribution scheme, it is sufficient that each node knows one column of this matrix, so that a node k only needs to store s^k to be able to compute the column of interest to it.

To set up the scheme, a random $(\lambda + 1) \times (\lambda + 1)$ symmetric matrix D over $GF(q)$ is constructed and used to compute the $N \times (\lambda+1)$ matrix $A := (D \cdot G)^T$ with $(D \cdot G)^T$ being the transpose of matrix $(D \cdot G)$. The matrix D is to be kept secret from all adversaries and sensor nodes. Each sensor node will only learn one row of matrix A. Due to the fact that D is symmetric, $A \cdot G$ also is a symmetric matrix as

$$A \cdot G = (D \cdot G)^{\mathrm{T}} \cdot G = G^{\mathrm{T}} \cdot D^{\mathrm{T}} \cdot G = G^{\mathrm{T}} \cdot D \cdot G = G^{\mathrm{T}} \cdot A^{\mathrm{T}} = (A \cdot G)^{\mathrm{T}}$$

Let $K := A \cdot G$ be the resulting symmetric matrix, so we know that $K_{i,j} = K_{j,i}$. The values of this matrix represent the keys to be used between node i and node j. To be able to generate keys with arbitrary peer nodes, node k stores the kth row of matrix A and the kth column of matrix G or s^k, respectively. When two nodes i and j want to establish a pairwise secret key, they exchange their values s^i and s^j, respectively, and use the received column of matrix G together with their own row of matrix A to compute the element $K_{i,j} = K_{j,i}$ of matrix $A \cdot G$. This scheme is secure as long as any $\lambda + 1$ columns of matrix G are linearly independent and no more than λ rows of matrix A are known to the attacker [Blo85].

Du et al. have proposed a scheme in [DDHV03a] that combines Blom's scheme with the probabilistic key distribution idea of Eschenauer and Gligor. Instead of generating one random symmetric matrix D, they propose to generate ω matrices $D_1, D_2, \ldots, D_\omega$ and to randomly select τ matrices for each node and store one row of the corresponding matrices $(D_l \cdot G)^{\mathrm{T}}$ in the node. If two nodes i and j want to establish a pairwise key, they first exchange the indices of the matrices from which they have stored one row each. If they find a common index, they share a common key space and can establish a pairwise key according to Blom's scheme. As compromise of a sensor node only reveals one row of each matrix, an attacker first needs to compromise enough sensor nodes to obtain at least $\lambda + 1$ rows from one key space before he can calculate other keys from this key space.

The probability that an attacker succeeds in compromising one key space depends on the parameters ω and τ, and Du et al. give an analysis for this in [DDHV03a]. As in the original scheme, pairwise keys are suggested to be set up between neighboring nodes, however, ω and τ have to be chosen so

that the probability with which two neighboring nodes share a common key space is high enough. To choose appropriate values for these parameters, the relationships between probability of key compromise, memory consumption in sensor nodes, desired key-connectivity, and transmission range of sensor nodes have to be carefully taken into account.

In Ref. [HK04], Hwang and Kim analyze these relationships for the three probabilistic key distribution schemes by Eschenauer and Gligor [EG02], Chang et al. [CPS03], and Du et al. [DDHV03a] with the help of "random graph theory" orginally published by Erdös and Rényi in Ref. [ER60]. Based on their analysis, they propose to adjust the transmission range of sensor nodes during key establishment by temporarily increasing transmission power. Although using a different technical means of implementation (increasing transmission power), this idea points in a similar direction like the one of Du et al. to establish keys between two-hop neighbors and leveraging the gain in "(two-hop) node-connectivity" to reduce the probability of compromised key spaces by increasing ω [DDHV03a].

Another construction that also avoids compromise of keys as long as no more than λ shares of key distribution data are compromised was proposed by Blundo et al. in Ref. [BDSH$^+$92] and is generally referred to as "polynomial based key predistribution." The main idea is to distribute partially evaluated bivariate polynomials of degree λ to nodes. The coefficients of the polynomials are chosen from $GF(q)$ for a sufficiently large q and the polynomials are constructed to be symmetric, that is $P(x, y) = P(y, x)$. Each node i receives a polynomial share $f_i(y) = P(i, y)$. Nodes i and j can establish a pairwise key $K_{i,j}$ by evaluating $f(i, y)$ at point j, or $f(j, y)$ at point i, respectively. Liu and Ning have combined this scheme together with the basic idea of Eschenauer and Gligor in Ref. [LN03].

Key management for wireless sensor networks has been a quite active area of research during the last couple of years and many variations of the schemes discussed in this chapter as well as some further ideas have been proposed and analyzed. Good survey articles are Refs. [CY05,HWM$^+$06].

The article *Security for the Mythical Air-dropped Sensor Network* by Gamage et al. [GBCT06] adds a critical perspective to the discussion. Based on an experimental evaluation of the average radio reception range of low-cost sensor nodes placed at two different antenna heights, 1 m above ground (to simulate manual installation) and sensor on the ground (to simulate random deployment by airplanes, which is an often referenced motivation, especially for most work on probabilistic key management), the authors critically discuss the assumptions made in many research papers and draw conclusions questioning the practical relevance of the proposed approaches. The tests were carried out in three different environments to simulate a forest (area with trees and bushes), a desert (open space with minimum above ground structure), and an urban alleyway (long wide corridor with moving people and several static objects). The most interesting observations from the experiments were that in open space the average radio reception range dropped sharply from 35 m (at 1 m above ground) to 7 m (at ground level), and that altitude had relatively small influence in the corridor tests (42–35 m). Considering the best case for air dropping of 10,000 sensor nodes on to a grid of 100×100 sensor nodes on a desert, e.g., the resulting sensor network would cover a maximum of $490,000\,\text{m}^2 \sim 0.5\,\text{km}^2$ (with 100,000 sensor nodes, the covered area would still be less than $5\,\text{km}^2$). From these considerations, the authors draw the conclusion that for effective and reliable coverage in applications such as battlefield monitoring, a sensor network needs to have a very high node density, requiring a high amount of sensor nodes and thus making the nodes very vulnerable to detection and capture by attackers. Based on this argumentation, the authors of Ref. [GBCT06] question the appropriateness of the assumptions underlying many publications and promote the use of simpler schemes, e.g., as proposed in the previously described SPINS approach [PST$^+$02].

One further open issue regarding key management in sensor networks stems from the fact that in many of the proposed approaches, keys of compromised nodes are supposed to be revoked. However, how to detect compromised nodes still is a problem that has not yet been solved to a sufficient degree, even though some first approaches have been proposed [SMR$^+$05,RZL06].

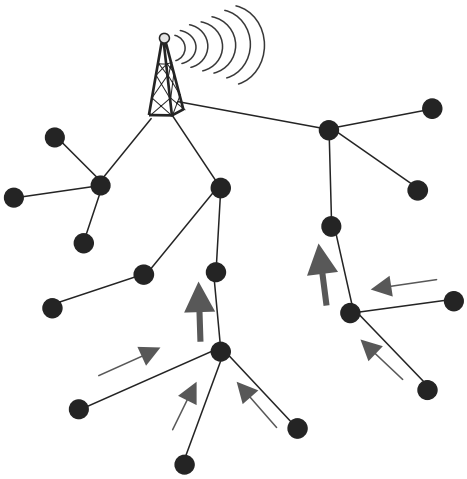


FIGURE 10.8 Aggregating data in a sensor network.

10.6 Secure Data Aggregation

As already mentioned in the introduction, data from different sensors is supposed to be aggregated on its way toward the base station (see also Figure 10.8). This raises the two questions, how to ensure authenticity and integrity of aggregated data, and how to ensure confidentiality of data to be aggregated? Concerning authentication and integrity of data reported toward the base station, if every sensor would add an MAC to its answer to ensure data origin authentication, all (answer, MAC)-tuples would have to be sent to the base station to enable checking of the authenticity. This shows that individual MACs are not suitable for data aggregation. However, if only the aggregating node added one MAC, a subverted node could send arbitrary data regardless of the data sent by sensors.

At GlobeCom'03, Du et al. proposed a scheme [DDHV03b] that allows a base station to "check the integrity" of an aggregated value based on endorsements provided by so-called witness nodes. The basic idea of this scheme is that multiple nodes perform data aggregation and compute an MAC over their result. This requires individual keys between each node and the base station. To allow for aggregated sending of data, some nodes act as so-called data fusion nodes, aggregating sensor data and sending it toward the base station. As a data fusion node could be a subverted or malicious node, its result needs to be endorsed by witness nodes. For this, neighboring nodes receiving the same sensor readings compute their own aggregated result, compute an MAC over this result and send it to the data fusion node. The data fusion node computes an MAC over its own result and sends it together with all received MACs to the base station. Figure 10.9 illustrates this approach.

In more detail, the scheme is described in Ref. [DDHV03b] as follows:

- Sensor nodes $S_1, S_2, ..., S_n$ collect data from their environment and make binary decisions $b_1, b_2, ..., b_n$ (e.g., fire detected) based on some detection rules.
- Every sensor node sends its decision to the data fusion node F, which computes an aggregated decision SF.
- Neighboring witness nodes $w_1, w_2, ..., w_m$ also receive the sensor readings and compute their own fusion results $s_1, s_2, ..., s_m$. Every w_i computes a message authentication code MAC_i with key k_i it shares with the base station, $MAC_i := h(s_i, w_i, k_i)$, and sends it to the base station.

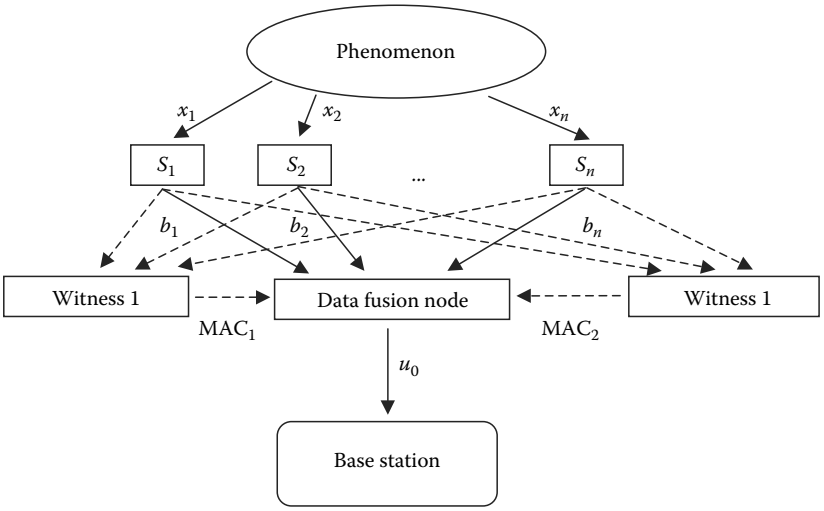


FIGURE 10.9 Overview of the witness-based approach [DDHV03b].

- Concerning the verification at the base station, Du et al. propose two variants. The first one is an $m+1$ out of $m+1$ voting scheme and works as follows:
  1. Data fusion node F computes its MAC:
  $$MAC_F := h(SF, F, k_F, MAC_1 \oplus MAC_2 \oplus \cdots \oplus MAC_m).$$
  2. F sends to base station: $(SF, F, w_1, ..., w_m, MAC_F)$.
  3. Base station computes all $MAC'_i = h(SF, w_i, k_i)$ and the authentication code to be expected from F:
  $$MAC'_F := h(SF, F, k_F, MAC'_1 \oplus MAC'_2 \oplus \cdots \oplus MAC'_m).$$
  The base station then checks if $MAC'_F = MAC_F$ and otherwise discards the message.

  If the set $(w_1, ..., w_m)$ remains unchanged, the identifiers of the w_i need only to be transmitted with the first MAC_F to save transmission bandwidth. There is, however, one major drawback with this scheme: If one witness deliberately sends a wrong MAC_i, the aggregated data gets refused by the base station (representing a DoS vulnerability).
- In order to overcome the DoS vulnerability of the first scheme, Du et al. also propose an n out of $m+1$ voting scheme:
  1. F sends to the base station: $(SF, F, MAC_F, w_1, MAC_1, ..., w_m, MAC_m)$.
  2. Base station checks if at least n out of $m+1$ MACs match, that is at least $n-1$ MAC_i match MAC_F.

  This scheme is more robust against erroneous or malicious witness nodes, but requires a higher communication overhead as m MACs must be sent to the base station

In Ref. [DDHV03b], Du et al. analyze the minimum length of the MACs to ensure a certain tolerance probability $2^{-\delta}$ that an invalid result is accepted by a base station. For this, they assume that each MAC has the length k, there are m witnesses, no witness colludes with F and F needs to guess the

endorsements MAC_i for at least $n-1$ witnesses. As the probability of correctly guessing one MAC_i is $p = 1/2^k$, the authors compute the chance of correctly guessing at least $n-1$ values to

$$P_S = \sum_{i=n-1}^{m} \binom{m}{i} p^i (1-p)^{m-i}$$

After some computation they yield:

$$m(k/2-1) \geq \delta$$

From this, Du et al. conclude that it is sufficient if $mk \geq 2(\delta + m)$, and give an example how to apply this. If $\delta = 10$ so that the probability of accepting an invalid result is 1/1024, and there are $m = 4$ witnesses, k should be chosen so that $k \geq 7$. This observation is supposed to enable economizing transmission effort.

In case that a data fusion node is corrupted, Du et al. propose to obtain a result as follows: If the verification at the base station fails, the base station is supposed to poll witness stations as data fusion nodes, and to continue trying until the n out of $m+1$ scheme described above succeeds. Furthermore, the expected number of polling messages $T(m+1, n)$ to be transmitted before the base station receives a valid result is computed.

Regarding the security of the proposed scheme, however, it has to be considered if an attacker actually needs to guess MACs to send an invalid result? As all messages are transmitted in clear, an eavesdropper E could easily obtain valid message authentication codes $MAC_i = h(s_i, w_i, k_i)$. If E later on wants to act as a bogus data fusion node sending an (at this time) incorrect result s_i, it can replay MAC_i to support this value. As Ref. [DDHV03b] assumes a binary decision result, an attacker only needs to eavesdrop until it has received enough MAC_i supporting either value of s_i. Thus, the scheme fails completely to provide adequate protection against attackers forging witness endorsements.

The main reason for this vulnerability is the missing verification of the actuality of an MAC_i at the base station. One could imagine as a quick fix letting the base station regularly send out random numbers r_B that have to be included in the MAC computations. In such a scheme, every r_B should only be accepted for one result, requiring the generation and exchange of large random numbers. A potential alternative could make use of time stamps, which would require synchronized clocks.

However, there are more open issues with this scheme. For example, it is not clear what should happen if some witness nodes cannot receive enough readings? Also, it is not clear why the MAC_i are not sent directly from the witness nodes to the base station? This would at least allow for a direct n out of $m+1$ voting scheme, avoiding the polling procedure described above in case of a compromised data fusion node. Furthermore, the suffix mode MAC construction h(message, key) selected by the authors is considered to be vulnerable [MOV97, note 9.65].

A further issue is how to defend against an attacker flooding the network with "forged" MAC_i (forged meaning arbitrary garbage that looks like a MAC)? This would allow an attacker to launch a DoS attack as an honest fusion node could not know which values to choose. One more "hotfix" for this could be using a local MAC among neighbors to authenticate the MAC_i. Nevertheless, this would imply further requirements (e.g., shared keys among neighbors, replay protection), and the "improved scheme" nevertheless would not appear to be mature enough to rely on it.

Some more general conclusions that can be drawn from this are that first optimization (e.g., economizing on MAC size, message length) can be considered as being one of the attacker's best friends, and that second in security, we often learn (more) from failures. Nevertheless, the article of Du et al. allows to discuss the need and the difficulties of constructing a secure data aggregation scheme that does not consume too many resources and is efficient enough to be deployed in sensor networks. As such it can be considered as a useful contribution despite its security deficiencies.

In Ref. [Wag04], Wagner studied the problem of the influence compromised sensor nodes can have on an aggregated result in a more fundamental way. He assumes all n sensors to report their

readings x_i directly to the base station which will perform the aggregation function $f(x_1, \ldots, x_n)$. In case that an attacker succeeds to compromise up to k of these sensor readings (e.g., by compromising sensor nodes, or manipulating their environment regarding the parameter they are measuring), it is depending on the aggregation function f how big the influence α is the attacker can have, where α is a multiplicative factor describing how much the manipulated aggregated value differs from the aggregated value in case of no attack. The article describes a mathematical model that is based on estimation theory. The main findings obtained from this model are that sum, average, minimum, and maximum are insecure estimation functions, that the average with values being only included in the aggregation computation if they lie in an a priori known range $[l, u]$ is problematic, and that the 5%-trimmed average (ignoring the lowest and highest 5% of the readings) performs better, as long as no more than 5% of the nodes are compromised. The most resistant aggregation function turns out to be the median, as it is secure as long as less than 50% of the nodes are compromised.

Even though Wagner only studied the problem under the assumption that all nodes directly report to the base station (securely based on an individual-shared key between each sensor node and the base station), these results in fact represent rather bad news for all aspirations to perform data aggregation with data origin authentication and integrity assurance. First, it turns out that unfortunately the more robust aggregation functions cannot be efficiently computed in a distributed procedure. Second, if all sensor readings being processed in an aggregation step were to be authenticated with an MAC, aggregation would not be able to significantly reduce the amount of data to be transmitted to the base station. Therefore, the compromise of aggregation nodes will give potential attackers an increased influence on the final aggregated value, as it allows them to manipulate a higher number k of sensor readings.

Concerning confidentiality of sensor readings to be aggregated on their way toward the base station, various approaches have been proposed that are based on "privacy homomorphisms" [RAD78]. A privacy homomorphism is an encryption transformation that allows direct computation on encrypted data. Let Q and R denote two rings, + denote addition and × denote multiplication on both rings, and let K denote the set of potential encryption keys. An encryption transformation $E : K \times Q \rightarrow R$ and the corresponding decryption transformation $D : K \times R \rightarrow Q$ are called a privacy homomorphism with respect to +, or ×, respectively, if given $a, b \in Q$ and $k \in K$ it holds:

$$a + b = D_k(E_k(a) + E_k(b))$$
$$a \times b = D_k(E_k(a) \times E_k(b))$$

In Ref. [GSW04] Girao et al. proposed a "concealed data aggregation (CDA)" scheme that deploys Domingo-Ferrer's additive and multiplicative privacy homomorphism [Jos96], which is based on modular exponentiation. The scheme allows to perform additive and multiplicative aggregation on encrypted data as it flows toward the base station and to extract the aggregated value at the base station. The operations to be computed for this at encrypting sensor nodes include modular exponentiation, so that this scheme reveals to be rather resource consuming for data sources. Another drawback of the scheme is that it requires a network-wide encryption key to be known by all sensor nodes, so that the compromise of a single sensor node is sufficient to defeat the scheme. As furthermore, Wagner has shown in Ref. [Wag03] that Domingo-Ferrer's privacy homomorphism is insecure for some major parameter settings, the scheme cannot be recommended for practical use.

Inspired by the work of Girao et al., various alternative approaches were proposed [HLN+07, RKP07,CMT05]. The approach of He et al. [HLN+07] is based on the additive property of polynomials as well as basic linear algebra (matrix inversion and multiplication), and it enables cluster-based joined summation of sensor readings while preserving privacy of individual sensor readings as long as less than $n-1$ nodes of a cluster of size n collude. In the same paper, a second scheme is proposed that achieves confidentiality of individual sensor readings by partitioning them in J individual pieces, that is representing sensor reading d_i of node i as a sum of J partial terms $x_i = \sum_{j=1}^{J} d_{i,j}$, and sending

$J-1$ of the partial terms encrypted to a different node of a randomly chosen set of $J-1$ nodes. All nodes in the network wait a certain period before they sum up all received partial terms together with their own share, and aggregate the readings on their way toward the base station. While this scheme is able to withstand up to $J-2$ colluding attackers before confidentiality is compromised, it gives individual compromised nodes extensive opportunity to manipulate the final aggregated value, as no integrity or plausibility check is provided for. Also, it increases the amount of messages to be sent and processed by a factor of J.

Ren et al. propose a privacy homomorphism-based data aggregation scheme ensuring data confidentiality combined with hop-by-hop MACs based on elliptic curve cryptography to provide basic integrity protection for exchanged messages [RKP07]. Because the same privacy homomorphism is used as in Ref. [GSW04], the same security concerns regarding compromised nodes apply with respect to confidentiality protection. Also, with its combination of the Domingo-Ferrer privacy homomorphism and the hop-by-hop computation of MACs, the approach puts a rather heavy load on participating sensor nodes.

To achieve confidentiality-preserving data aggregation in sensor networks without incurring computation-intensive operations, Castellucia et al. propose a rather simple scheme that is based on modular addition [CMT05]. Each sensor node i shares a secret key with the base station and uses this key to generate pseudorandom keystream sequences k_i that are used to encrypt individual sensor readings x_i. Furthermore, all sensor nodes know a network-wide parameter M, where M is chosen large enough so that later operations will not lead to an overflow (see below). To encrypt a sensor reading x_i, each node computes $E(x_i, k_i, M) = x_i + k_i \bmod M$. To aggregate two encrypted messages, they are simply added modulo M. When the base station receives all aggregated values together with set I of identifiers of the nodes that contributed to the aggregated values it computes $\sum_{i\in I} E(x_i, k_i, M) - \sum_{i\in I} k_i \bmod M$. To be able to generate the correct keystreams k_i, some kind of synchronization between sensor nodes and base station is required (e.g., a counter that is transmitted together with the sensor node's identifier). If n sensor nodes may contribute to an aggregated value, M needs to be chosen so that the addition of the n sensor readings will not overflow. Therefore, M should be chosen so that $M \geq 2^{\lceil n\cdot\log_2(\max(x_i))\rceil}$ with $\max(x_i)$ representing the maximum possible value for a sensor reading x_i. As the scheme relies on individual keys between sensor nodes and the base station, the compromise of individual sensor nodes does not affect the confidentiality of other node's sensor readings. Furthermore, the authors recommend to combine this scheme with hop-by-hop authentication to avoid external attackers to be able to inject bogus sensor readings, and also point out that such measures are likely not suitable to protect against compromised sensor nodes injecting false data (compare also the discussion of Ref. [Wag04] above). The main advantages of this approach are its low computational overhead and the increased security resulting from individual keys per sensor node. Two major drawbacks are that the base station needs to know which node contributed to an aggregated value, and that the use of keystreams needs to be synchronized between sensor nodes and the base station, both limiting the total amount of data transmissions that can be economized by this scheme.

10.7 Summary

Wireless sensor networks are an upcoming technology with a wide range of promising applications. As in other networks, however, security is crucial for any serious application. Prevalent security objectives in wireless sensor networks are confidentiality and integrity of data, as well as availability of sensor network services being threatened by DoS attacks, attacks on routing, etc. Severe resource constraints in terms of memory, time and energy, and an "unfair" power balance between attackers and sensor nodes make attaining these security objectives particularly challenging. Approaches proposed for wireless ad hoc networks which are based on asymmetric cryptography are generally considered to

be too resource consuming. This chapter has reviewed basic considerations on protection against DoS and attacks on routing, and has given an overview of first approaches proposed thus far. For ensuring confidentiality and integrity of data, the *SNEP* and μ*TESLA* protocols were discussed, and considering key management the LEAP protocol, as well as probabilistic key management, and its many variations have been reviewed. Designing security functions suitable for the specific communication patterns in sensor networks turns out to be rather difficult. Regarding data origin authentication and integrity, there seem to be some principle obstacles in achieving the goals of energy saving data aggregation and security at the same time [Wag04]. Concerning confidentiality of sensor readings to be aggregated on their way toward the base station, a number of approaches for CDA based on privacy homomorphisms [RAD78] have been presented and discussed.

References

[BDSH+92] Blundo, C.; De Santis, A.; Herzberg, A.; Kutten, S.; Vaccaro, U.; Yung, M.: Perfectly-secure key distribution for dynamic conferences. In: *Advances in Cryptology — CRYPTO'92, Proceedings of the 12th Annual International Cryptology Conference*, Santa Barbara, CA, Springer, August 1992, Lecture Notes in Computer Science, Vol. 740.

[Blo85] Blom, R.: An optimal class of symmetric key generation systems. In: *Proceedings of the EUROCRYPT'84 Workshop on Advances in Cryptology: Theory and Application of Cryptographic Techniques*. New York: Springer-Verlag, Inc., 1985. ISBN 0-387-16076-0, pp. 335–338.

[BR96] Baldwin, R.; Rivest, R.: *The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms*. October 1996. RFC 2040, IETF, Status: Informational, ftp://ftp.internic.net/rfc/rfc2040.txt

[CMT05] Castelluccia, C.; Mykletun, E.; Tsudik, G.: Efficient aggregation of encrypted data in wireless sensor networks. In: *Proceedings of the Second Annual International Conference on Mobile and Ubiquitous Systems (MobiQuitous 2005)*, 2005. San Diego, CA, pp. 109–117.

[CPS03] Chan, H.; Perrig, A.; Song, D.: Random key predistribution schemes for sensor networks. In: *Proceedings of the IEEE Symposium on Security and Privacy*, 2003. Oakland, CA.

[CY05] Camtepe, S. A.; Yener, B.: *Key Distribution Mechanisms for Wireless Sensor Networks: A Survey*, 2005. Technical Report TR-05-07, Rensselaer Polytechnic Institute, Computer Science Department, Troy, NY.

[DDHV03a] Du, W.; Deng, J.; Han, Y.; Varshney, P.: A pairwise key pre-distribution scheme for wireless sensor networks. In: *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, 2003. Washington, DC.

[DDHV03b] Du, W.; Deng, J.; Han, Y.; Varshney, P.: A witness-based approach for data fusion assurance in wireless sensor networks. In: *Proceedings of the IEEE 2003 Global Communications Conference (Globecom'2003)*, 2003. San Francisco, CA, pp. 1435–1439.

[DH76] Diffie, W.; Hellman, M. E.: New directions in cryptography. *Transactions on IEEE Information Theory IT-22* (1976), 644–654.

[EG02] Eschenauer, L.; Gligor, V. D.: A key management scheme for distributed sensor networks. In: *Proceedings of CCS'02*, 2002. Washington, DC.

[ElG85] ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* 31(4), (1985), 469–472.

[ER60] Erdös, P.; Rényi, A.: On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Science* 5 (1960), pp. 17–61.

[GBCT06] Gamage, C.; Bicakci, K.; Crispo, B.; Tanenbaum, A. S.: Security for the mythical air-dropped sensor network. In: *Proceedings of the 11th IEEE Symposium on Computers*

and Communications (ISCC'06). Washington, DC: IEEE Computer Society, 2006. ISBN 0-7695-2588-1, pp. 41–47.

[GNY90] Gong, L.; Needham, R. M.; Yahalom, R.: Reasoning about belief in cryptographic protocols. In: *Symposium on Research in Security and Privacy*. IEEE Computer Society, IEEE Computer Society Press, May 1990, pp. 234–248.

[GSW04] Girao, J.; Schneider, M.; Westhoff, D.: CDA: Concealed data aggregation in wireless sensor networks. In: *ACM Workshop on Wireless Security*. Philadelphia, PA, October 2004. Poster presentation.

[HD05] Huang, C.; Du, D.: New constructions on broadcast encryption and key pre-distribution schemes. In: *Proceedings of the IEEE INFOCOM'05*, 2005. Miami, FL.

[HK04] Hwang, J.; Kim, Y.: Revisiting random key pre-distribution schemes for wireless sensor networks. In: *SASN '04: Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks*. New York: ACM, 2004. ISBN 1-58113-972-1, pp. 43–52.

[HLN+07] He, W; Liu, X.; Nguyen, H.; Nahrstedt, K.; Abdelzaher, T. T.: PDA: Privacy-preserving data aggregation in wireless sensor networks. In: *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM'2007)*, 2007. Anchorage, AK, ISSN 0743-166X, pp. 2045–2053.

[HMNS98] Haller, N.; Metz, C.; Nesser, P.; Straw, M.: *A One-Time Password System*. February 1998. RFC 2289, IETF, Status: Draft Standard, ftp://ftp.internic.net/rfc/rfc2289.txt

[HPJ02] Hu, Y.; Perrig, A.; Johnson, D.: *Wormhole Detection in Wireless Ad Hoc Networks*. July 2002. Technical Report TR01-384, Rice University, Houston, TX.

[HWM+06] Hegland, A. M.; Winjum, E.; Mjolsnes, S.F.; Rong, C.; Kure, O.; Spilling, P.: A survey of key management in ad hoc networks. *Communications Surveys and Tutorials* 8(3), (2006), 48–66. IEEE.

[Jos96] Domingo i Ferrer, J.: A new privacy homomorphism and applications. *Information Processing Letters* 60(5), (1996), 277–282.

[KW03a] Karl, H.; Willig, A.: *A Short Survey of Wireless Sensor Networks*. 2003. TKN Technical Report Series, TKN-03-018, Technical University Berlin, Germany.

[KW03b] Karlof, C.; Wagner, D.: Secure routing in wireless sensor networks: Attacks and countermeasures. *AdHoc Networks Journal* 1(2–3), (2003), 293–315.

[KW05] Karl, H.; Willig, A.: *Protocols and Architectures for Wireless Sensor Networks*. Chichester, U.K.: John Wiley & Sons, 2005. ISBN 0470095105.

[KY98] Kaliski, B. S.; Yin, Y. L.: *On the Security of the RC5 Encryption Algorithm*. 1998. RSA Laboratories Technical Report, TR-602, Version 1.0.

[LN03] Liu, D.; Ning, P.: Establishing pairwise keys in distributed sensor networks. In: *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, 2003. Washington, DC.

[LN04] Liu, D.; Ning, P.: Multilevel μTESLA: Broadcast authentication for distributed sensor networks. *Transactions on Embedded Computing Systems* 3(4), (2004), 800–836. ISSN 1539–9087.

[MOV97] Menezes, A.; van Oorschot, P.; Vanstone, S.: *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press LLC, 1997.

[MS77] MacWilliams, F. J.; Sloane, N. J. A.: *The Theory of Error-Correcting Codes*. Amsterdam the Netherlands: Elsevier Science Publishing Company, Inc., 1977.

[OR87] Otway, D.; Rees, O.: Efficient and timely mutual authentication. *Operating Systems Review* 21(1), (1987), 8–10.

[PLGP06] Parno, B.; Luk, M.; Gaustad, E.; Perrig, A.: Secure sensor network routing: A clean-slate approach. In: *Conference on Future Networking Technologies (CoNEXT)*, 2006. Lisboa, Portugal.

[PST+02] Perrig, A.; Szewcyk, R.; Tygar, J.; Wen, V.; Culler, D.: SPINS: Security protocols for sensor networks. *Wireless Networks* 8 (2002), 521–534.

[PT03] Perrig, A.; Tygar, J. D.: *Secure Broadcast Communication in Wired and Wireless Networks*. Norwell, MA: Kluwer Academic Publishers, 2003.

[RAD78] Rivest, R. L.; Adleman, L.; Dertouzos, M. L.: On data banks and privacy homomorphisms. *Foundations of Secure Computation*. Academic Press, (1978), 169–179.

[RKP07] Ren, S. Q.; Kim, D. S.; Park, J. S.: A secure data aggregation scheme for wireless sensor networks. In: *ISPA 2007 Workshops*, Springer, 2007. Lecture Notes in Computer Science, Vol. 4743, pp. 32–40.

[RSA78] Rivest, R.; Shamir, A.; Adleman, L.: A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM* 21(2), February 1978, 120–126.

[RZL06] Roman, R.; Zhou, J.; Lopez, J.: Applying intrusion detection systems to wireless sensor networks. In: *Proceedings of the 3rd Consumer Communications and Networking Conference (CCNC'2006)*, 2006. Las Vegas, NV.

[Sch03] Schäfer, G.: *Security in Fixed and Wireless Networks*. New York: John Wiley & Sons, 2003.

[Sch05] Schaefer, G.: Sensor network security. In: Zurawski, R. (ed.): *The Embedded Systems Handbook*. Boca Raton, FL: CRC Press, 2005, pp. 39.1–439.23.

[SMR+05] da Silva, A. P. R.; Martins, M. H. T.; Rocha, B. P. S.; Loureiro, A. A. F.; Ruiz, L. B.; Wong, H. C.: Decentralized intrusion detection in wireless sensor networks. In: *Proceedings of the 1st ACM International Workshop on Quality of Service and Security in Wireless and Mobile Networks (Q2SWinet'05)*. New York: ACM, 2005. ISBN 1-59593-241-0, pp. 16–23.

[TAL01] Tuomas Aura, P. N.; Leiwo, J.: DOS-resistant authentication with client puzzles. In: *Proceedings of the Security Protocols Workshop 2000*, Springer, 2001. Cambridge, U.K., April 2000, Lecture Notes in Computer Science (LNCS).

[Wag03] Wagner, D.: Cryptanalysis of an algebraic privacy homomorphism. In: *Proceedings of the Sixth Information Security Conference (ISC'03)*, Springer, October 2003. Lecture Notes in Computer Science, Vol. 2851.

[Wag04] Wagner, D.: Resilient aggregation in sensor networks. In: *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'04)*, New York: ACM Press, 2004, pp. 78–87.

[Woo01] Wood, A.: *Security in Sensor Networks*. 2001. Sensor Networks Seminar, University of Virginia, VA.

[WS02] Wood, A.; Stankovic, J.: Denial of service in sensor networks. *IEEE Computer* 35 (October, 2002), 54–62.

[ZSJ03] Zhu, S.; Setia, S.; Jajodia, S.: LEAP: Efficient security mechanisms for large-scale distributed sensor networks. In: *Proceedings of CCS'03*, 2003. Washington, DC.

11

Wireless Sensor Networks Testing and Validation

11.1 Introduction 11-1
Testing in the Context of Wireless Sensor Networks • A Case for Coordinated Testing and Validation
11.2 Wireless Sensor Network Validation 11-5
Wireless Sensor Network Test Platforms • Software Testing Methodologies
11.3 Sensor Network Testbeds 11-9
Deployment-Support Network
11.4 Integrated Testing Architecture 11-10
Influence of the Environment • Architecture Overview • Continuous Integration • Continuous Integration in the Sensor Network Context • Testbed Integration • Physical Parameter Extraction • Physical Stimulation • Test and Instrumentation Architecture for Physical Characterization
11.5 Test and Validation for Life on the Glacier—The PermaSense Case 11-21
PermaSense System Architecture • PermaSense—Testing and Evaluation
11.6 Summary 11-25
References 11-25

Matthias Woehrle
Swiss Federal Institute of Technology Zurich

Jan Beutel
Swiss Federal Institute of Technology Zurich

Lothar Thiele
Swiss Federal Institute of Technology Zurich

11.1 Introduction

Since Kahn et al.'s [1] vision of "Smart Dust" where minuscule sensors are deployed without particular effort, e.g., dropped from a plane and instrumenting areas at large scale, wireless sensor networks (WSNs) have matured into viable systems for sensing the environment for diverse applications.

Nevertheless, the wealth and complexity of issues on the system level often only unraveling to their full magnitude in practice, have led to substantial efforts being made in areas unanticipated by the early visionaries. As an example, experience has shown that exact sensor node placement is highly critical not only for sensing but also for wireless communication and the scavenging of energy from the environment. Details often deemed trivial such as node placement [2], node protection (e.g., to protect against bird droppings [3]), or wireless propagation closely over ground [4] have rendered system design and development an extremely complex and often error-prone process requiring careful planning.

WSNs are expected to become a radical innovation similar to the internet. For a widespread adoption, the underlying architecture needs to be reliable. Best-effort approaches with superior service levels as used in mobile telephony need to guarantee basic reliability and quality of service to provide

satisfiable results to the end user. Nevertheless, the design of a WSN system with its complex interactions and system intricacies today requires in-depth knowledge and care for detail. As the field of WSN continues to open up new application spaces, interaction and interdisciplinarity has to be supported. Roemer et al. [5] provide a broad overview of applications of these wireless networked embedded systems to present a comprehensive design space of WSNs. The diversity in application areas results in differing requirements for the system in terms of reliability, availability, quality of sensor data, latency constraints, energy-efficiency, and longevity.

A framework has to be provided to determine detailed requirements of a platform, in order to be able to aggressively optimize the system and provide users with a usable and satisfactory system solution. The level of detail for accessing, programming, or optimizing system characteristics needs to be different for a wide range of expertise of the system users.

WSN technology as many embedded systems has a long lifetime without the capabilities of easy updates in the field. Thus, functional correctness at deployment time is of utmost importance. However, numerous WSNs deployments fail to perform [6,7] or to work at all [8], albeit ingenious engineering efforts. A critical factor is that due to the novel environments and tight system integration within, detailed models necessary for an understanding of the requirements are not available. As an example in environmental monitoring, seasonal changes and its effect on plant growth heavily influence sensed data, harvestable energy, and the properties of communication [9]. This results in many system designs having to rely on either no, weak or simplified assumptions. The realism of failing and underperforming deployments has heavily influenced system design. Detailed provisions and focused debug, maintenance and monitoring enhancements improve WSN reliability and performance. Researchers try to actively attack these problems by integrating sophisticated mechanisms into their designs relying on in-depth knowledge [10] and increased visibility by design [11]. Various tools attack the debugging at the deployment site by listening to WSN traffic [12], remote access to the sensor nodes [13], visual monitoring [14], or integrating source-level debugging mechanisms for remote debugging [15]. Nevertheless, WSN design today is tedious and requires attention to intricate details.

In order to deploy correct and performing systems, WSNs require tools for verification and validation. Although validation in the form of testing can only determine the presence of errors and not their absence, testing is the primary method employed in software, hardware, and system development. Thus, we present testing tools and methodologies in the context of wireless networked embedded systems, detail its peculiarities and explain according challenges.

11.1.1 Testing in the Context of Wireless Sensor Networks

WSNs integrate very different characteristics from various, previously unrelated fields: they are distributed systems built of embedded sensor nodes with wireless communication. Validation and testing methodologies thus need to address the domain-specific intricacies and challenges.

11.1.1.1 Device Constraints

The sensor nodes have significantly benefited from Moore's law integrating minuscule microprocessors, memory and radio chips at a reasonable price. Looking at the future there seem to be two trends concerning computing and shrinking feature sizes: scale-up with small mote-class devices in large numbers and scale-out with larger devices allowing for more functionality.

Larger devices allow for more ease-of-use in design and fit well in tiered architectures. Not all applications require aggressive energy optimization and in select cases may work with powerful platforms, especially when possibilities for energy harvesting [16] are abundant. However cost and power considerations in large-scale deployments favors mote-class devices, which require intricate attention in design and testing.

Due to the tight resource constraints of embedded systems, software components and resource usage have to be rigorously optimized. One of the most fundamental issues is the constraint of energy

supply. For such heavily energy constrained systems, power consumption needs to be meticulously minimized. Since power consumption is such a pivotal factor for a WSN system, the integration of tests especially targeted for power consumption is required throughout the whole design and development process. Code size is another critical factor as program memory is limited. Compile time checks integrated into a automated build framework address these issues. Another vital aspect of the test architecture is visualizing the trends of program memory usage as individual software components are added to the design.

11.1.1.2 Communication Intricacies

Radio communication is a characterizing trait of WSNs. It introduces unreliability in message transmissions and incurs significant power consumption. Unreliable communication requires each node to incorporate countermeasures for failed transmissions due to interferences and collisions. The broadcast nature of the medium necessitates arbitration of the broadcasting nodes to avoid collisions. Noise, anisotropies, fading, and multi-path effects deteriorate packet reception rates.

A particular aspect of typical WSNs is that computation is cheaper than communication, i.e., $E_{\text{comp}} \ll E_{\text{comm}}$. Thus processing of data fundamentally consumes less power than communication of data [17,18]: While computation is not for free, sending 1 kb of information across 100 m consumes the amount of energy required for executing millions of instructions on a general-purpose microprocessor. In general, energy for transmitting and receiving are comparable in power consumption. Even idle listening is comparable in its current draw, necessitating coordination in transmission to avoid wasting energy. As communication is expensive in terms of energy costs, a general goal is to minimize the amount of messages to be transmitted or received. A customized design must consider trade-offs concerning the accuracy, periodicity, and latency requirements of the sensor data.

Considering the small amount of payload the sensor nodes typically transmit, the control overhead concerning the maintenance and protocols are significant. Additionally, debug or health information as well as design for testing mechanisms have to be carefully selected as they can have a significant impact on these low data transport applications. However, the protocol stack itself can benefit from additional visibility [11]. There is an inherent trade-off between the observability of a system and its efficiency in the use of resources. A system spending the majority of the time in a sleep state or accessible only by a highly bandwidth-limited and unreliable wireless link is challenging to observe, which is a requirement for the successful analysis and identification of causes of error.

Execution information typically does not allow for standard analysis of the communication as the interpretation heavily depends on customized, nonstandardized protocols and the instrumentation of the system [19]. As an example, individual message transactions may be observed by logging $n_{\text{send}} = k$ sending attempts. Due to lost messages, missed acknowledges and retransmission attempts, the number of according reception successes $n_{\text{succ}} \in [0, k]$. However, instrumentation on the application level may return just a single notification of transmission success on each the sender and the receiver for multiple send and receive events on the medium access control (MAC) layer. Although the same action has occurred, the event traces and their interpretation differ considerably.

11.1.1.3 Distributed System State

In WSNs, system state is distributed over the sensor nodes and messages. Sensor nodes are highly concurrent and only loosely coupled through unreliable, wireless communication. Testing of a comprehensive WSN system includes the checking of predicates on the distributed, global state. Observing global state via local instrumentation or passive inspection raises issues of consistency of snapshots and causality [20]. Concurrent and thus partial ordered actions in between synchronization events, produce a substantial number of linear extensions, each a valid representation of system execution. Probe effects are aggravated as the probed device and its context in the system are perturbed. Thus, the same test inputs may result in different outputs for individual runs.

11.1.1.4 Tight Integration

Real WSN deployments are tightly integrated into the environment rendering them extremely susceptible to the effects of harsh and considerably varying environmental conditions. This integration is a commonly known cause of faults. Haneveld [9] reports unanticipated obstructions: The leaves of the plants present on the deployment site changed transmission characteristics when there was rain. He observed a correlation between packet reception ratio, temperature, and humidity. In the initial Great Duck Island deployment Szewzcyk et al. [21] discuss node failures and attributes a number of them to humidity and broken seals on the packaging. The impact of environmental conditions may be deeply rooted as in the case of the heavily used TI MSP430 [22]. It is a microcontroller directly designed for ultralow power system design. Power mode changes take effect immediately. Nevertheless, even such a targeted device needs to be handled with care for long-term outdoor deployments: One example is the negative temperature coefficient of the digitally controlled oscillator (DCO) when the DCO is disabled for long low-power sleep states. In outdoor deployments, e.g., relying on scavenging energy from the sun, a system may be put to sleep for prolonged periods of time during the night. Such a long-term sleep period can result in a delayed wakeup of the DCO as the device is impacted by a variation in temperature. The effect starts to severely show with sleep periods of tens of minutes and a temperature variation of tens of degrees, which are common characteristics in environmental monitoring. This and similar effects are documented in the fine print of the manufacturer documentation but often overlooked in practice.

11.1.2 A Case for Coordinated Testing and Validation

The design of WSNs is hard: Failed and unsuccessful deployments [7] underline the need for (a) better design tools and methodologies and (b) for an increased focus on the validation of WSNs.

Some of the challenges of WSNs are

- Large state space of the sensor nodes due to interrupt-driven software with preemptive scheduling
- Vast distributed state space of highly concurrent sensor nodes
- Dynamically varying spatial locality due to unreliable, broadcast medium
- Visibility of the system is limited and expensive, since data traffic is low and energy scarce
- In-depth optimization of resource consumption
- Tight integration into environment increases susceptibility and dependencies on environment
- Integration of issues and challenges from differing fields (wireless, embedded, mechanical, and concurrent) into an integrated design methodology

Therefore, the testing of WSNs must encompass various aspects of the system under development. Tests and testing tools need to consider the intricacies of the embedded devices combined with the complexities of testing a distributed system. In addition, the unreliable and heavily bandwidth constrained radio communication aggravates these issues. Thus validation and testing are considerably difficult and complex.

The following section presents WSN validation using testing techniques and test platforms specifically designed for the sensor network case. A classification of test platforms is provided to illustrate the applicability of a platform for certain classes of tests. A prominent and widely used test platform is a WSN testbed, which is presented in Section 11.3 using the deployment-support network (DSN) as an example. Finally, a test architecture is presented which integrates established methodologies to address the common validation issues in WSN development.

11.2 Wireless Sensor Network Validation

The focal point of the design process is a specification, collecting all functional and nonfunctional requirements of the system. It is the core document during the development and used to verify and validate the systems function and performance. It has been shown that efforts spent on specification and the identification of the requirements yields better results over the whole system development life cycle [23]. The specification as the core document is the fundamental basis for validating WSN systems.

Especially for interdisciplinary research between other scientific fields such as geosciences and the WSN community, there is a significant challenge in building functioning, performing, and reliable systems. The reason is that the system may not be completely specified at design time. Researchers typically do not know the quantity of required sensors, the sampling period, nor do they have a clear and definite description of the system environment. Thus researchers look for the culmination of an ultimate sensor node for an initial prototype: feature-rich, everlasting, resilient, and reliable. Before restricting to limited data researchers are looking for a rich data set to better understand the observed phenomenon and to be able to derive an accurate model. Some projects have already shown that prototyping is a viable approach [24]. Starting from an over-provisioned, feature-rich prototype, collected real-world data may be used to minimize sensory inputs [25]. Some technologies like the Sun Spots [26] combine a more powerful processor with a low-power microcontroller for controlling the duty cycles for deep sleep states. This might be a first step toward a prototyping platform, since it allows to employ the standard Java design methodology along with available tools and support.

In the following, we focus on validating a comprehensive WSN system by applying testing strategies. However, formal verification tools and methods are invaluable to assist the validation of system components.

11.2.1 Wireless Sensor Network Test Platforms

The design of a WSN systems relies on using different test platforms which allow for testing different system properties and functionalities. Figure 11.1 compares different test platforms along three characteristic properties: scalability, visibility, and abstraction. The following sections describe the individual properties, present prominent examples and a discussion of the application areas of individual tool classes.

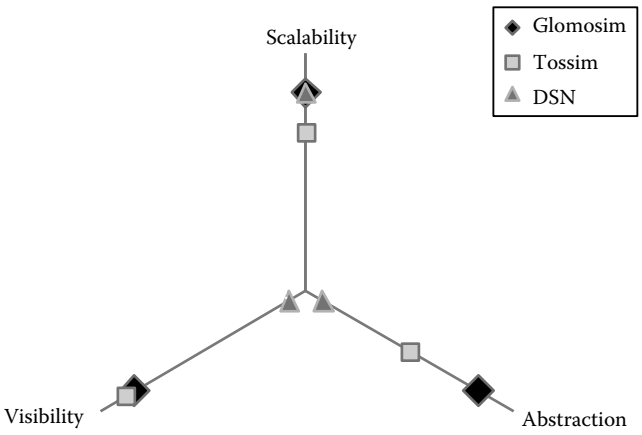


FIGURE 11.1 Comparison of different test platforms along the three major test platform properties.

11.2.1.1 Characteristics of WSN Test Platform

1. *Scale*
   WSN systems range from small installments with a dozen to hundreds of nodes. Scale refers to the capability of the test platform to capture the system in operation including sensor node instantiations and models of the communication channels in between and the environment the system is embedded into at scale. There are two different sources for scaling issues: technical limitations and cost. While simulation allows for tests in large quantities, testbed installations using sensor nodes have been limited due to the considerable costs. Detailed simulation suffers from substantial computational requirements, which necessitates simulations on a higher abstraction level in order to cope with this scaling complexity as shown in Figure 11.2.
   Validation of WSNs needs to consider widely varying test areas. In the context of scale, these can be classified into local problems, direct neighborhood, and global problems. Global problems may further be divided into subsets according to cluster, path, tree, or system problems.
   Local problems on a single sensor node concern traditional testing of embedded system with strict resource limitations, e.g., driver testing for hardware interfaces such as an I2C bus. With two nodes, communication on a single link can be tested. Three nodes allow for testing communication with collisions and hidden-terminal effects as well as a primitive topology for routing: Multihop algorithms and protocols can be tested using a predefined topology by setting a fixed neighborhood table. For testing effects of dense or complex topologies, route reestablishment, and effects of partitions and node unavailability a larger number of nodes are required.
2. *Visibility*
   Visibility describes the concept of the availability of information on the internal state of the devices and their environment at any time in the test execution. Embedded systems typically have a very limited visibility. The bandwidth to the system is limited, because it is

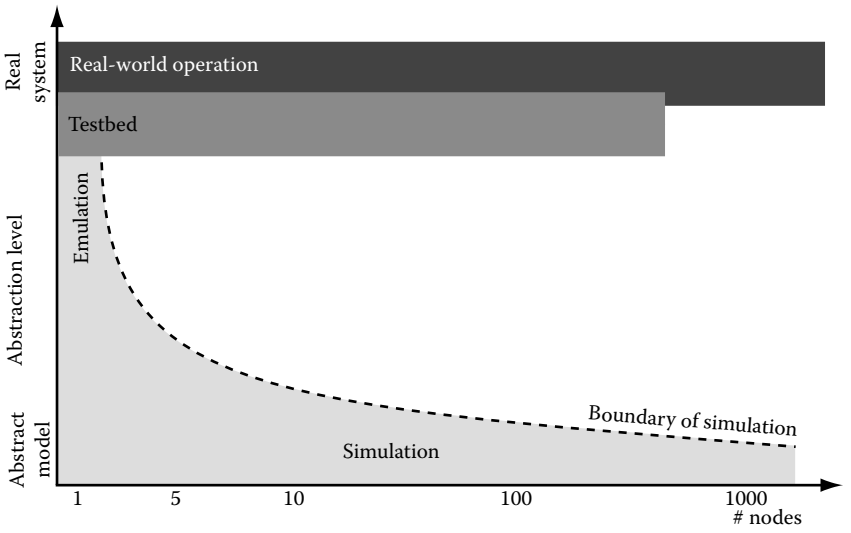


FIGURE 11.2 Simulation allows for efficient scaling but is bounded by the models and abstractions used. Testbeds and real deployments are closer to reality but considerably more difficult to handle or simply not as accessible, therfore scaling is limited.

optimized for its operational use in order to minimize costs. For sensor nodes, the interfaces are limited to the radio, a serial interface, individual pins, and some LEDs. Radio and LEDs are expensive in their power consumption, limiting their use in real deployments with batteries. In order to avoid probe effects [27], using the radio for inspection should be avoided during testing the system. Passive inspection [12] allowing for monitoring a system without perturbation is limited in its applicability in testing as fails cannot be attributed to the system as the platform itself cannot provide any guarantees on a reliability. Off-line gathering of data collected in unused memory during the test is a viable option. However, memory access can interfere radio communication when they share a bus, e.g., on the Tmote Sky Platform. LEDs merely allow for primitive instantaneous checking.

Simulation tools allow for detailed inspection of the system. The monitoring is reliable and does not perturb the system under test. Platforms offering high visibility allow for extensive logging to find correlation between log events. Substantial data gathered with TOSSIM [28] allows for diagnostic simulation using data mining techniques [29]. For testing on real sensor nodes with lower visibility, monitoring data is focussed to detect merely a specific error or failure. Monitoring data volume is highly test-specific: Testing for reboots under environmental conditions merely requires a single bit, while tests concerning distributed protocols such as for routing incur a substantial logging amount.

3. *Abstraction*

   Testing the system in its final deployment location and its ultimate scale is typically prohibitive. Thus, test platforms have to abstract from the final system and build models for the testing (cf. Figure 11.2). The models include device, network, communication, and environmental abstractions.

   (a) Device

   While execution on a sensor node and tests using an instruction-level simulator use the comprehensive compiled application, simulation tools typically abstract from the device. TOSSIM uses actual NesC code, but provides simulation libraries for hardware-dependent code. Other discrete event simulators such as GloMoSim [30] or Castalia [31] are platform and operating system agnostic, nevertheless allowing for modeling comprehensive communication models.

   (b) Network and communication

   Network topology and radio communication have a substantial impact on a sensor network. Wireless communication over low-power radios is varying in time and distance. The result is transitional, asymmetric, and irregular links. This is exacerbated when the topologies become complex and obstructions create multipath and fading effects. This fact has a considerable impact on the protocol stack. When simulating sensor networks great care must be taken to accustom for this considerable impact. Models must be finely tuned to represent an actual deployment. Overly simplistic models and assumptions [4] hide the complexity and nondeterministic behavior found in actual WSN deployments.

   (c) Environment and energy

   Sensor nodes are deeply embedded with its environment. As such a deployment is heavily dependent on its environment: Temperature and humidity affect node operation and communication. Solar energy to be harvested and battery drainage heavily affect longevity and sustainability. Simulators typically do not consider these effects. Indoor testbeds may incorporate special instruments to allow for controlling battery profiles of a power supply. Temperature effects for alpine environments can be simulated by using climate chambers.

Programming on a low-level fosters type errors and requires hardware interaction necessitating close attention to timing, resource arbitration, and the interrupt strategy. Detailed understanding of the hardware is important, including susceptibility to environmental conditions, long sleep cycles, and the architecture of the sensor node as a composition of interacting components. Traditional embedded test and debugging systems may be used for a single node: Emulators such as Avrora or execution on a real node and exposing internal state via a serial interface. Debugging help is provided by tools such as JTAG or Marionette [13]. For hardware-independent problems, simulators such as TOSSIM can be used. The details of modeling of wireless communication heavily influence the protocol stack. Asymmetry of links, anisotropy, and noise in the environment drastically influence protocol performance. As communication is optimized for efficient energy use, the optimization must rely on valid assumptions and models of the actual deployment environment.

11.2.1.2 Example Instances of Test Platforms

The test platforms used in WSN validation include well-known simulators with their origin in traditional (wireless) network design like Ns-2 [32] or GloMoSim [30]. Libraries have been developed by the community to include the domain-specific aspects of WSN, particularly the radio. They offer scalability, but since they were not designed for WSN, they typically provide a higher ease-of-use and accuracy for other domains than for WSN. Simulation libraries for discrete event simulators, especially targeted for WSNs, such as Castalia [31] based on OMNeT++, provide a sensor node agnostic test platform. Such libraries are targeted for validation of sensornet algorithms. WSN-specific simulators have been established, like EmSim [33] and TOSSIM [28], which make use of the actual target code and link it with simulation libraries to run on a host computer for simulation.

Simulation deficiencies and realism of deployment challenges has triggered an increased interest in testbed implementations such as Motelab [34] or the DSN [35]. These typically are installations at the department building or on the university campus to better grasp deployment characteristics and the nondeterministic nature of the wireless communication.

Other significant parameters in the system space, such as solar energy scavenged for sustainable operations and others effect of a outdoor environment, require either physical stimulation and control or a representative outdoor testbed such as Trio [3]. Nevertheless, testbed characteristics for radio communication, topology, or harvestable energy may still differ considerably to the actual deployment site.

Researches have proposed field trials [36] in order to attain deeper insight into WSN deployments. Turau et al. discuss that such prototypical deployments are expensive, but nevertheless provide invaluable information about the environment and actual system execution on a large-scale, especially as systematic verification and testing methodologies and tools are still in their infancy.

Each individual test platform has its benefits concerning a specific design under test, design stage, and a set of tests. A distinct approach is integration of tools: Hybrid solutions (EmStar [33]) offer simulation, emulation, and testbed execution in a single framework. COOJA [37] allows for cross-level simulation, thereby leveraging the benefits of individual tools. A distributed testing framework for WSNs [38] supports continuous testing throughout the design cycle by exploiting the ability to design and test on different abstraction levels with a common test specification, promoting regression testing and test integration into a periodic build process.

11.2.2 Software Testing Methodologies

Design-by-contract [39] is an established methodology in software engineering. For embedded software, contracts on component interfaces help to determine faults in the form of contract violations.

In TinyOS, components which are connected through narrow interfaces defining usage and provision of its functions, profit from specifying interface contracts [40]. Contracts expose the component specification in an executable format and thus allow for automatic checking of correct interface usage. Contracts can also be used for unit testing of individual software components, e.g., in Avrora. A less formal, yet similar approach is nCUnit [41], a unit testing framework for the nesC language, which allows for formulating assertions on the test execution.

T-Unit [42] is a unit testing framework for TinyOS software, available as a contributed project from the TinyOS repository. Different from traditional unit testing platform, it extends basic functionality by allowing execution on one or more actual sensor nodes. Tests are controlled from a test host over the serial interface and contain all software required to run the functionality on the target hardware. This approach allows for characterization and testing of TinyOS software functionality on and across nodes, e.g., transmissions. Regression testing on the TinyOS core libraries underlines the significance of unit testing for validation.

Nguyen et al. [43] discuss program representation for TinyOS application in the form of an application posting graph (APG). The event-based nature of applications and preemptive scheduling of the event handlers renders the control flow of an application and thus the APG complex. Nevertheless, the APG allows for analyzing coverage of tests and facilitates structural testing.

11.3 Sensor Network Testbeds

Sensor network testbeds allow the execution of code on an actual target device, possibly even in a realistic environmental setting. With respect to other tools such as simulation and emulation this approach comes closest to reality, i.e., the actual sensor network deployment but also allows to actually execute the compiled code in its binary form. By doing so a number of effects that are either hidden away through the simplifying nature of abstractions and models used in simulations or are not showing to be of significance can be revealed to the developer. Such effects range from the correctly ordered execution of code to impairments of the wireless channel and the actual availability of resources and more.

Typical WSN testbeds that are in use today consist of a wired back-channel that allows to reprogram and log data from nodes distributed accross a larger space. Testbeds such as MoteLab [34] or TWIST [44] are typically set up in office environments and consist of tens to hundreds of nodes wired either using Ethernet or USB, sometimes a combination of both. In practical operation a software to be tested would be uploaded to the testbed, programmed onto all the nodes in the testbed and subsequently executed. The data logged from all the testbed nodes is made available at a central location for postexecution analysis. Different testbeds contain a number of different auxiliary features, e.g., user arbitration, scheduling of execution, instant notification, etc., however they are typically stand-alone and not integrated with other development tools.

11.3.1 Deployment-Support Network

The DSN [35] is a specialized testbed that uses a wireless back-channel and therefore can be quickly deployed in different locations, e.g., outdoors. Moreover it features a logical and physical separation of the testbed plane and the devices under test (DUT plane)(see Figure 11.3). With respect to testbeds like Motelab or TWIST, this separation into individual distributed observers and the nodes to be tested has the advantage of being able to customize test scenarios and integrate stimuli and asynchronous testing. Furthermore the decentralized augmentation with other test and measurement equipment is facilitated.

The DSN testbed is built from BTnodes (DSN nodes) and supports the connection of most popular Mote architectures using and interface board (e.g., Mica2, Tmote Sky, TinyNode, etc.). It is

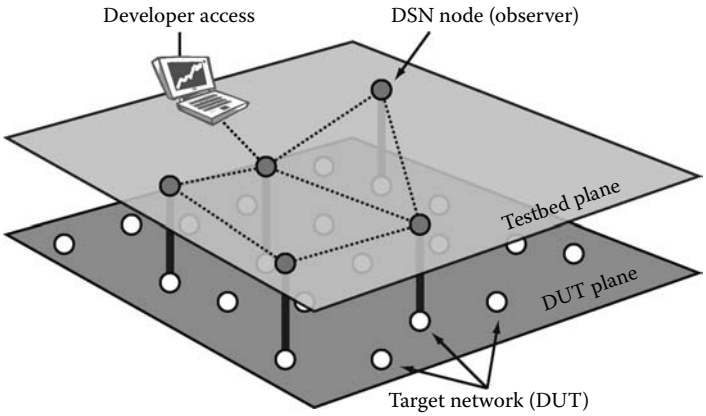


FIGURE 11.3 DSN is a WSN testbed that separates distributed observers (DSN nodes) and the target network under development or the DUT into two separate planes.

controlled from a central server and database component that presents all interaction, logs, and test results to the user. For successful logging by the testbed the software running on the nodes under test has to be augmented with test monitors that output all state and data to be logged to the observer nodes.

11.4 Integrated Testing Architecture

In order to extend the possibilities of current testbeds such as introduced in Section 11.3 we develop an integrated testing architecture that is geared at giving the sensor network designer the most transparency possible while optimally taking into account the circumstances and the environment the senor network is to be deployed in. In the following we present a test and instrumentation architecture that augments WSN testbeds by incorporating the environment and giving exact and detailed insight into the reaction to changing parameters and resource usage.

11.4.1 Influence of the Environment

The goals of designing and developing for correct function and in the case of WSNs, the efficient use of resource consumption are key to longevity and sustainability in sensor network applications. There is indeed a need for comprehensive and widely spread tools and methodology for system testing. It is important to not just simply create a single new tool but an integrated methodology taking into account analytical and empirical methods on different levels of abstraction [37,38].

Validation of the system with systematic testing strategies is an indispensable part to improve the development process in order to arrive at a functioning and performing system implementation.

Testing of WSN system needs to cover the complete system state space concerning the hardware resources and environment. Current testbeds are fixed at a single point although the system state space considering the environment and resource configuration is large, as depicted in Figure 11.4. However, it is vital to determine effects of a drained battery, cold temperature, or unprecedented obstructions to determine a functionally correct system under such operating conditions in the actual deployment. Thus our testing strategy is to provide various defined resource and environmental conditions to expose software faults, which are triggered by condition changes.

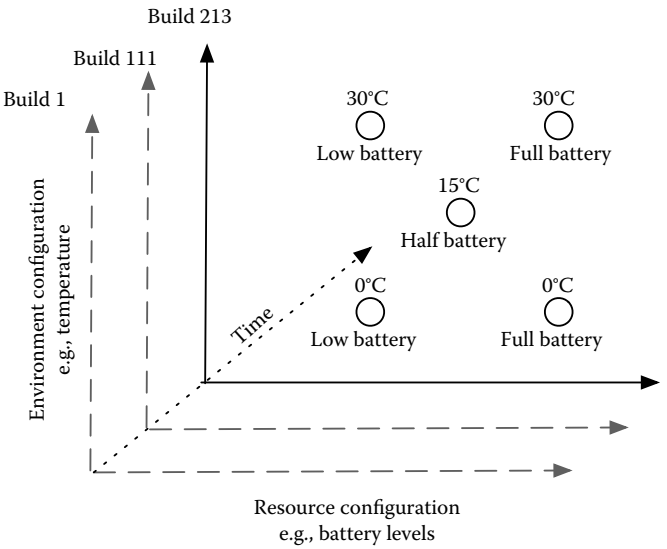


FIGURE 11.4 Environment and resource configuration can be controlled for automated, continuous testing of regression builds.

11.4.2 Architecture Overview

As depicted in Figure 11.5, the test and instrumentation architecture is composed of three basic components: a software repository, an infrastructure for continuous integration (CI), and a testing infrastructure. The software repository provides a central location for all software under development and associated tests. It allows to keep track of the development progress and allows interaction of multiple users.

The core of the testing architecture is the infrastructure for CI. While CI is an established method and tool commonly used in enterprise software engineering [45], we propose to integrate it with a specialized testing infrastructure to account for the peculiarities of WSN development—execution on a real embedded target and in a distributed testbed environment [34,35] with tools for hardware testing and stimulation. The CI server regularly checks for changes in the project software repository and upon detection of a change initiates a clean checkout and then builds the project. The extension with a testing infrastructure allows to deploy and execute the software under test directly on the specific hardware platform it is being designed for. The tight integration with the test infrastructure allows to monitor the test and provides detailed information about functional and physical details of the execution on real target nodes. All data monitored during the test is collected and logged into a central database referencing the exact software version in the repository that was used to generate the respective instance.

In the following we will provide the details on the building blocks used in our testing architecture with a special focus on the integration of the physical test, i.e., the test on real motes in a real environment. The mechanics and equipment used in our prototypical setup is briefly explained in the following.

11.4.3 Continuous Integration

CI [45] is a methodology, which promotes frequent integration, i.e., after each code check-in, in order to provide rapid feedback to developers. This facilitates identifying software defects,

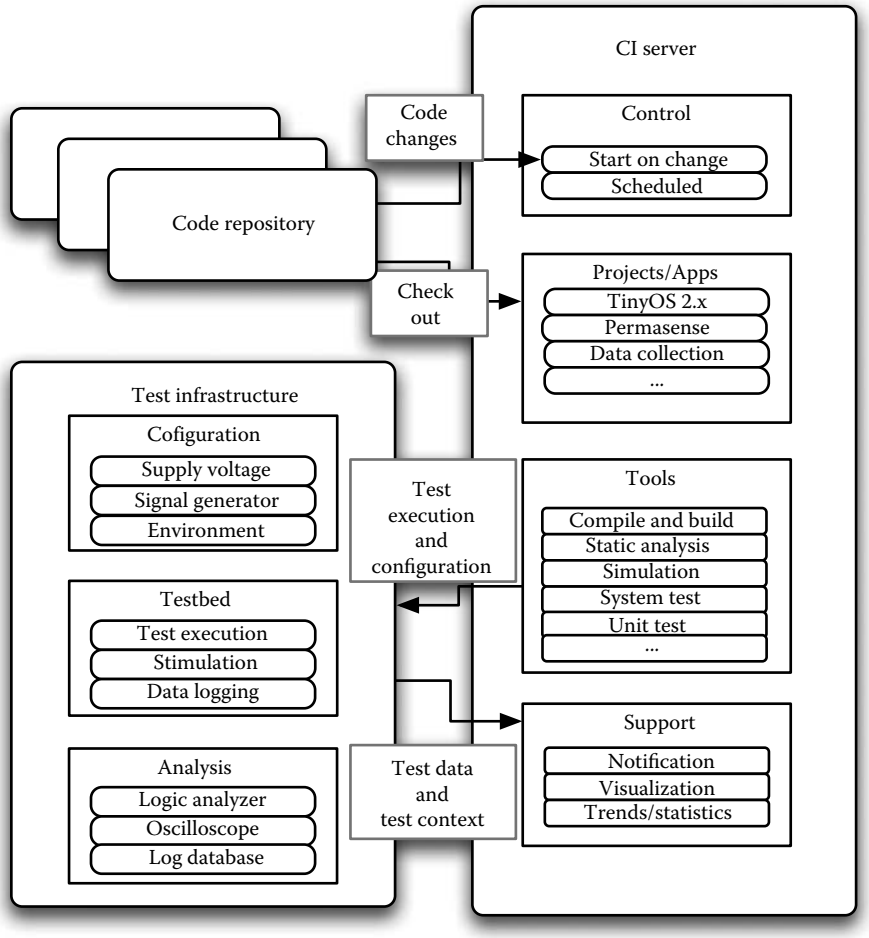


FIGURE 11.5 Overview of the test and instrumentation architecture.

since differences are miniscule and error sources are easier to pinpoint, typically subject to a recent change. Up to this point, CI focusses on the integration of enterprise scale software projects designed by large teams and is a common and well-known methodology, e.g., in agile development [46].

For integration of software in a team, CI allows for communication with the code repository, a build tool-chain to compile the software, software analysis tools, and a test platform running unit tests such as JUnit for Java projects. The overall status of the project as well as the details of all associated builds are presented on a Webpage, often also using a central indicator such as a lava lamp. Each build shows the software status providing visibility to the team. A prominent example of a CI server is [47], a java-based, thus OS-independent and mature tool with an extensive user base. CruiseControl features a Web interface, which displays all available projects. The configuration of CruiseControl is performed in a single XML file. This includes a description of each of the projects including information on repository monitoring, scheduling of builds, logging, and publishing of results. The build process of a project is configured in a separate XML file unique to each project. Each individual project features a Webpage displaying information on the recent builds and access to published build

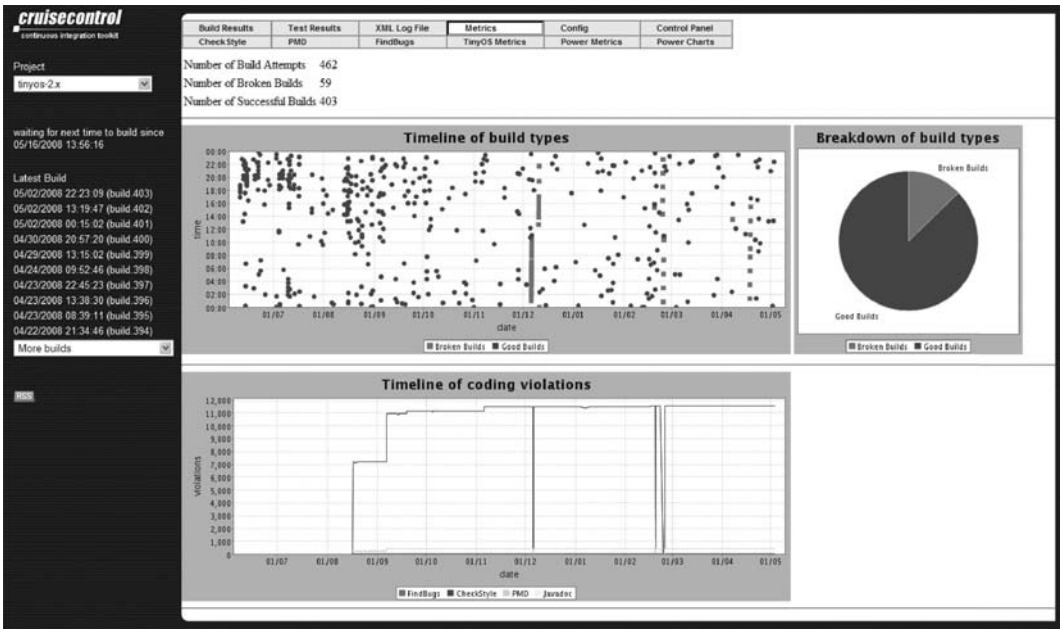


FIGURE 11.6 Build result status, histogram, and statistics are reported on the CruiseControl Webpage.

data. CruiseControl allows for integrating widgets and build tools and works with third party tools such as Trac.

11.4.3.1 Regular Builds with Statistics

Regular builds are either triggered upon code changes introduced to the repository, by user requests using a Web interface or using periodic timers. Builds can be configured separately allowing for usage of individual parameters and tools to be used. Builds are referenced to a specific version of the codebase in the repository using a unique build id. All data generated, logs, build artifacts, and test results are stored in a central data structure available through a Web-based reporting interface (Figure 11.6).

11.4.3.2 Source Code Analysis and Code Checkers

Different tools for checking, e.g., coding (style) violations such as Findbugs, PMB, CheckStyle, or the correctness of javadoc documentation, are readily available for integration into common CI frameworks such as CruiseControl. Recently, first specialized tools to check WSN-specific software emerge such as the contract checker for TinyOS components [40]. Checking and analysis tools run during or after the build process and generate reports that can be visualized using the CruiseControl reporting front-end. However they are typically tools allowing static code analysis only, without actually executing the binary generated and thus cannot detect all defects in the code, nor the handling of the software of boundary conditions concerning interrupts, sensor values, or hardware-related problems concerning the battery level or the environmental conditions of the device.

11.4.3.3 Unit Test Integration

Unit testing is a methodology focussed on testing small components, in particular the smallest components available such as individual functions or methods. A test wraps around the code under test and allows to execute the unit isolated given some preconditions and a guaranteed behavior inside

a test harness. This allows to check for the correctness of function of code written and persistence over a longer period of development with often complicated integration. This divides the problem of checking for correct functionality into smaller pieces, is extremely popular in programming (e.g., JUnit [48]) and recently also available for TinyOS [42] and the nesC language [41]. It is widely applicable especially for agile methods such as test-driven development. Unit testing is a powerful tool, yet not sufficient to test the comprehensive system, as the system is more than just the union of its components. Complex interactions induced by the inherent parallelism and nondeterminism in distributed systems require additional testing on the complete system functionality, e.g., to assure that asserted preconditions hold. As our test architecture follows an integrative approach, unit testing can easily be integrated and ameliorate the coverage of tests.

11.4.3.4 Memory Profiling

Embedded system design revolves around the efficient usage of resources and correct functionality. Memory usage analysis is therefore a very important indicator of the status of a project. A simple, yet powerful test extension is a histogram of the memory usage (see Figure 11.14) and to test on a known upper bound of a specific platform, e.g., 48 kb in the case of an MSP430F1611 [22].

11.4.3.5 Notification on Build Failures

CI typically contains a notification mechanism. In the case of CruiseControl, these can be configured, e.g., to send email notification to all developers involved upon a failed build. These email notifications contain complete coverage of build status and the error logs in question. Repetitive notifications can be used to emphasize urgency and as a result are powerful mechanism to successful progress in teams. The results are anonymous and offer an objective way of telling everyone what went wrong based on a reference installation of both the toolchain and codebase. This eliminates defects stemming from individual developers computer setups, e.g., developers forgetting to include libraries that they use locally. Apart from the technical perspective, the social implication of this mechanism are obvious: No more personal confrontations of fellow developers neither across institutional borders nor in close vicinity of a shared office that are detrimental to productivity. Anonymity in this case is very healthy since it provides objective, qualitative, and quantitative measures (Figure 11.7).

11.4.3.6 Graphical Reporting

The features presented in the previous sections all rely on a tool running on some input and providing output data, generating reports as they are executed. While the original logs and text files generated are important when digging deep into details, a powerful graphical reporting interface helps to present an overview of the most critical aspects from the wealth of information and contexts. This provides the benefit to allow for depicting long periods of project development including dependencies in a concise representation. In some cases, as for the memory histograms shown in Figure 11.14 such an overview reveals trending information, thus aiding in the interpretation by providing powerful visual feedback.

11.4.4 Continuous Integration in the Sensor Network Context

The concept of regression builds, unit testing, and test automation are not new and already very successful in the enterprise software world. In this context, Java or similar programming languages prevail and software is typically built for and run on large servers. This is a software-focussed approach that does not take into account the hardware intricacies in deeply embedded systems, such as the details of the sensor node hardware platform or the influence of the environment. This is completely different to WSNs. The distributed and embedded nature of today's WSN systems does not allow for mapping the aforementioned procedures and tools directly from enterprise scale CI.

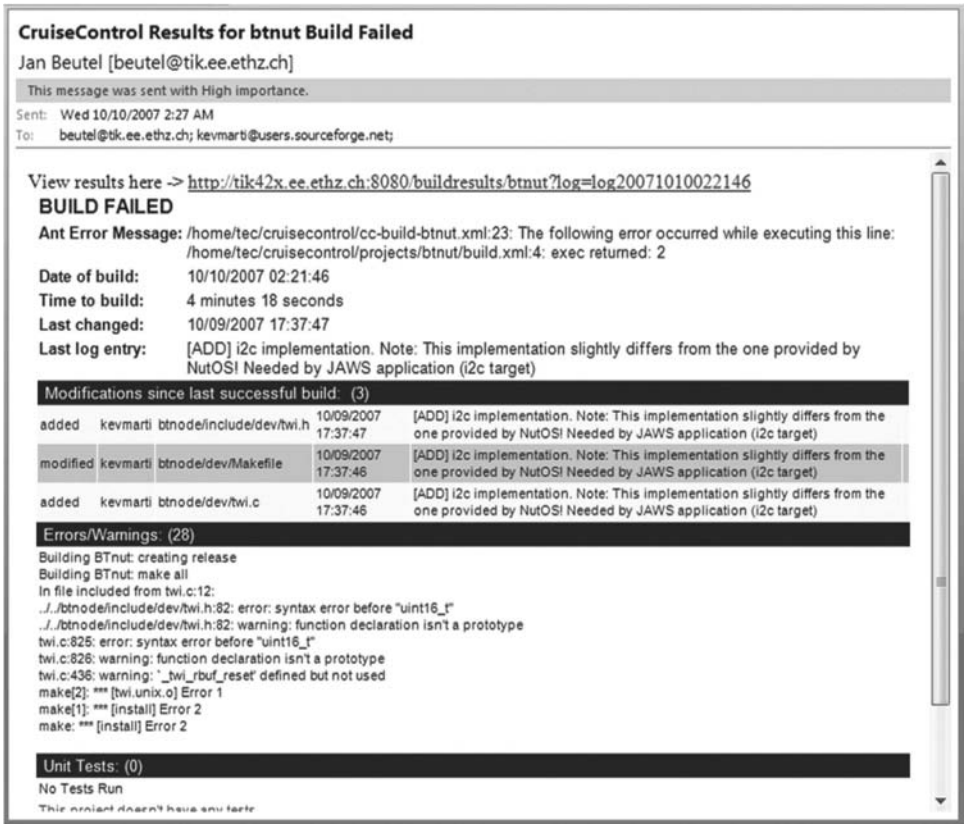

CruiseControl Results for btnut Build Failed
Jan Beutel [beutel@tik.ee.ethz.ch]
This message was sent with High importance.
Sent: Wed 10/10/2007 2:27 AM
To: beutel@tik.ee.ethz.ch; kevmarti@users.sourceforge.net;
View results here -> http://tik42x.ee.ethz.ch:8080/buildresults/btnut?log=log20071010022146
BUILD FAILED
Ant Error Message: /home/tec/cruisecontrol/cc-build-btnut.xml:23: The following error occurred while executing this line:
/home/tec/cruisecontrol/projects/btnut/build.xml:4: exec returned: 2
Date of build: 10/10/2007 02:21:46
Time to build: 4 minutes 18 seconds
Last changed: 10/09/2007 17:37:47
Last log entry: [ADD] i2c implementation. Note: This implementation slightly differs from the one provided by NutOS! Needed by JAWS application (i2c target)
Modifications since last successful build: (3)
added kevmarti btnode/include/dev/twi.h 10/09/2007 17:37:47 [ADD] i2c implementation. Note: This implementation slightly differs from the one provided by NutOS! Needed by JAWS application (i2c target)
modified kevmarti btnode/dev/Makefile 10/09/2007 17:37:46 [ADD] i2c implementation. Note: This implementation slightly differs from the one provided by NutOS! Needed by JAWS application (i2c target)
added kevmarti btnode/dev/twi.c 10/09/2007 17:37:46 [ADD] i2c implementation. Note: This implementation slightly differs from the one provided by NutOS! Needed by JAWS application (i2c target)
Errors/Warnings: (28)
Building BTnut: creating release
Building BTnut: make all
In file included from twi.c:12:
../../btnode/include/dev/twi.h:82: error: syntax error before "uint16_t"
../../btnode/include/dev/twi.h:82: warning: function declaration isn't a prototype
twi.c:825: error: syntax error before "uint16_t"
twi.c:826: warning: function declaration isn't a prototype
twi.c:436: warning: '_twi_rbuf_reset' defined but not used
make[2]: *** [twi.unix.o] Error 1
make[1]: *** [install] Error 2
make: *** [install] Error 2
Unit Tests: (0)
No Tests Run

FIGURE 11.7 Build results and error messages are directly reported using email notification.

Missing are the possibility to execute build results (executables) on a real WSN target device, e.g., a mote, to run distributed tests using a testbed and emulating the environment.

To this end, current CI practices are extended with the goal to provide a comprehensive development support especially targeted for wireless networked embedded systems. Thus, software builds are deployed on a real WSN testbed. Moreover, WSN testbeds extensions should be added to establish a testing infrastructure, which provides capabilities to capture physical parameters, instrumentation to capture logical data and allow for stimulation of the software under test. As depicted in Figure 11.4, testing for WSNs needs to provide control of environment and resource configuration. In order to guarantee test coverage and to be able to analyze test executions, the infrastructure has to allow for monitoring and controlling the resources available to the sensor nodes.

Testbeds allow for testing applications distributedly on a considerable scale. Further extensions to control the environment allow for a test setup, which closely matches the actual deployment conditions targeted. Combined with instrumentation to monitor physical parameters and resources available on the devices under test in the testbed, this is a powerful combination allowing to analyze the data extracted and compare it in real-time against expected results.

11.4.5 Testbed Integration

Our approach for testing WSN software combines established methods such as regression testing and unit testing with execution in a real environment. By using a testbed and realistic deployments

scenarios, many factors that cannot be captured in simulation and analysis methods, can be used to narrow in on a realistic sensor network deployment and simultaneously extract behavioral trace data from the system under test. Such a testbed can be located locally or even be integrated remotely. Even federated testbeds, each with specific operating environment characteristics or dedicated to a certain project or design team are feasible. Test jobs consist of a compilation process that is handled by the CI framework with some additional instrumentation: (1) job formation with subsequent submission to a testbed, (2) distribution of code to target devices, (3) synchronous start of all target devices, and (4) log file collection. Depending on the context of the test, analysis can take place online, e.g., for the monitoring of operation or in more detail off-line after completion of a test job.

11.4.5.1 Data Collection—Basic Logging

As already discussed in Section 11.4.3, CI and regression testing based on changes of the software codebase allows for gradual refinement. This allows to continuously check the status and progress of the integrated base of a project. Using continuous testing strategies, a design point that has been reached once and tested to satisfaction will not be lost again. In combination with the execution on a testbed additional tasks can be performed such as the variation and tuning of parameters (sensitivity analysis) or the variation of the network topology by using different subsets of nodes or even different platforms in the testbed. An example is the parametrized project of a low-power gathering applications, where the effects on changes on different sleep intervals are visualized as shown in Figure 11.8.

Without very elaborate customization or the incorporation of complicated tool setups, the integration of CI and a testbed allows to test, repetitively using a number of different analysis methods. Through the integrated approach however, execution is greatly simplified and data from all test jobs is logged in a repository that references the actual software code version under test. This assures a maximum of transparency and the ability to post facto analysis as the comprehensive system context is maintained alongside the test results. A widely neglected fact is that when testing is carried out manually, notes are taken in a lab notebook and log files are organized in individual, nonpersistent structures making it almost impossible to compare results from subsequent test runs. Test runs are a previous resource and must be maintained and organized for easy reference.

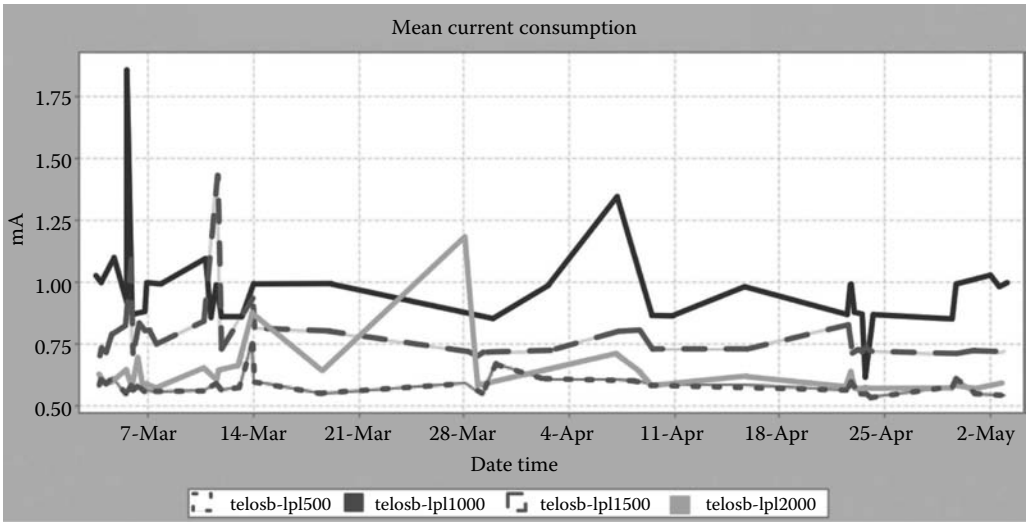


FIGURE 11.8 Parametrized data collection application project: plot of average power consumption for different sleep intervals.

11.4.5.2 Emulating the Environment

An apparent deficiency of current WSN testbeds is that they are typically set up in office environments and thus can only offer some realism with respect to the devices and an actual radio channel used. They still abstract from the real deployment as depicted in Figure 11.2. But they fail to capture the exact effects of a target deployment region, e.g., temperature variation, or changing operating conditions, e.g., depleting power resources. However these effects are important factors that have lead to a number of reported failures for real WSN deployments [7,9]. The effect of temperature on electronic devices is well known, often basically considered at design time but not elaborated in detail in a system context toward the end of a development phase. Since software running on WSN devices is rigorously optimized to the hardware, small changes in the operating environment can lead to a malfunction of the system. In a similar way the capacity of any battery is influenced both by the discharge behavior and the environmental conditions, e.g., the temperature. Today, this can neither be emulated nor estimated and requires testing on a live object.

We therefore suggest to further augment a testbed/CI architecture with means to emulate and control the environment in which nodes are deployed on a testbed. In practice this means that nodes under test are to be placed in a temperature and humidity cycling [temperature cycling test (TCT)] chamber and powered from different power sources, e.g., real batteries or preferably a programmable power source to emulate the behavior of a battery or a solar cell. This allows for testing the response of a system under test in both cyclic and boundary conditions, e.g., depletion of the battery. In combination with the ability of certain testbeds to insert asynchronous events [35], e.g., errors like a node failure or a delayed start of a subset of nodes, worst-case scenarios can be emulated on the system under test. Mixed scenarios are easily feasible, with a portion of nodes in a TCT chamber and a portion located outside subject only to the regular ambient conditions. Since in a mixed scenario the TCT chamber affect the propagation of the radio signals, we propose to situate all antennas of the devices under test outside of the TCT chamber using extension cabling. In cases where this is not feasible and yet weak signal conditions need to be emulated, (programmable) attenuators could be incorporated in the infrastructure. We have so far refrained from the increase in complexity and simply performed test runs alternating antennas both inside the TCT chamber and outside.

11.4.6 Physical Parameter Extraction

For the physical characterization of motes, we employ two different approaches: On the one hand we observe long-term trends to determine the development process effects on the characteristics. With detailed snapshots of individual software builds, we perform an in-depth analysis allowing for regression testing of physical parameters.

11.4.6.1 Long-Term Trending

Current testbeds only has means of profiling power consumption on select nodes [34]. High-precision distributed energy and power monitoring will be possible when tools such as SPOT [49] are available. Up to this point, due to the lack of inexpensive tools, widespread adoption has not been possible. A supervision of all nodes under test is mandatory for assuring reliable operation and sufficient test coverage. We are using a combination of a DSN [35] node pair with a custom power monitoring board (see Figure 11.9) that uses the internal analog-to-digital convertor (ADC) on the ATmega128l and a current sense amplifier in combination with the network logging tool Cacti. The DUT can be powered from different sources (battery, line power) and the power status is sampled on request once every 5 min. Although limited in precision and temporal resolution, basic long-term trending with coarse granularity is very helpful for testbed supervision on long test sequences, giving

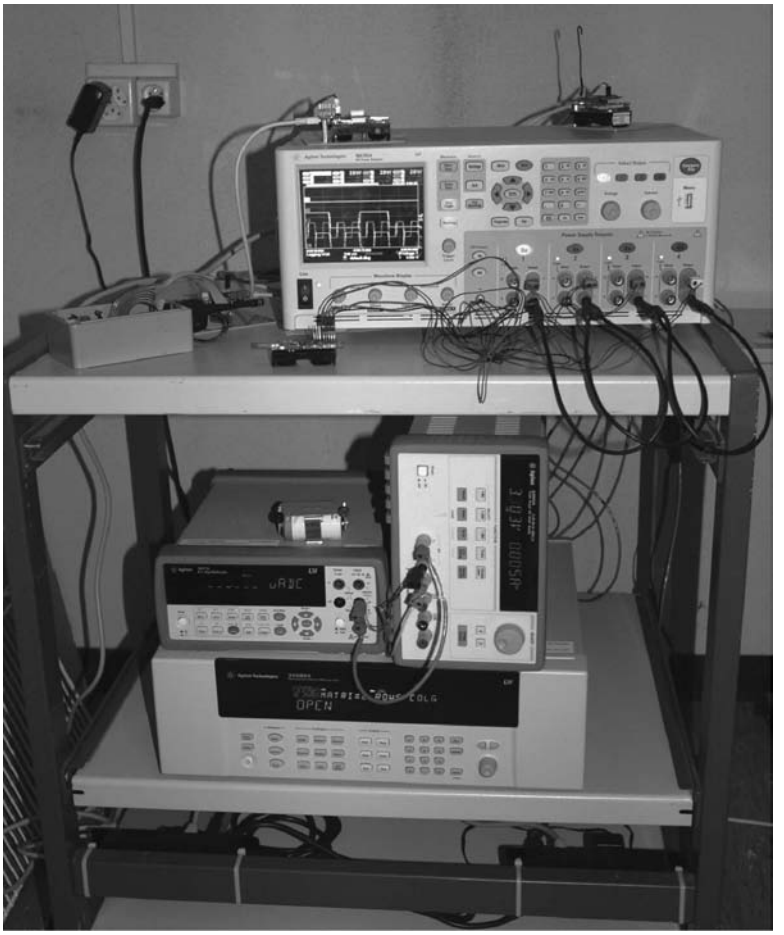


FIGURE 11.9 Testbed setup consists of a monitoring node, an interface board for power measurements, a supply, and the DUT.

an initial impression of "what is going on" (see Figure 11.13). Sampling more frequently and integrating the values would be valuable.

11.4.6.2 Detailed Physical Parameter Characterization

For detailed characterization of the system performance and especially to pinpoint specific behavior detailed power traces incorporating the variation of the input are of utmost importance. In order to characterize device variations, but also to understand the interplay of the power supply (battery, regulated power, and solar) and the system under varying load conditions instrumentation with a resolution >1000 samples/s is required. Since such equipment is currently both costly and bulky it is not feasible to instrument every node in every testbed. Therefore we propose to instrument only a number of nodes for detailed analysis and automate the process using an interface to CI.

11.4.6.3 Testing Dynamic Power Consumption Using Detailed Current Traces

As energy resources are very sparse for most sensor network deployments, it is of utmost importance to optimize for power consumption: Apart from average power consumption, the dynamic profile is important as it affects battery drain. Moreover, average power consumption over long test runs

misses spurious erratic behavior. The dynamic patterns observed in sensor nodes due to heavy duty cycling requires intricate analysis of power consumption.

For testing dynamic power consumption, oracles on power traces from individual nodes can be formulated. The oracles provide a boolean result, which is used in the CI framework. Testing of power traces is formulated on a given window of the test execution: Most test start with initialization, but power tests typically address steady-state operation. Hence, a certain offset for the power monitoring is required.

Basis for the oracle is a reference function: It is a function $f : \mathbb{R} \to \mathbb{R}$ of time t of a measured physical quantity, e.g., power consumption. $f(t)$ is a piecewise, typically discontinuous function, which is composed of subfunctions $f_i : \mathbb{R} \to \mathbb{R}$ with discontinuities $t_i \in \mathbb{R}$, $i \in \mathbb{N}$ at the subinterval boundaries of f. The reference function may be derived from a measured golden reference or by simulating a model. A model can be determined by different means, e.g., using the specification, i.e., data sheets, or linear regression model on previous measurements [50]. The sub intervals of $f(t)$ represent state changes of sensor node components.

From the reference trace, an acceptance region is determined: Measurements inside the acceptance region represent expected behavior. Single outliers may be measurement artifacts; for consecutive readings outside the acceptance region an error is asserted.

Figure 11.10 illustrates the two-step process of generating an upper and a lower bound based on a reference function. The first step is to consider differences in value range, which can be attributed to manufacturing variations or differing temperature levels: an offset of the measurement values in a specified subinterval is defined $\Delta y_i^{(+/-)}$. The reference function is hulled by two bounding functions $f_{y,i}^{(+/-)}$, which account for the variance in value in each interval by adding (subtracting) Δy_i^+ (Δy_i^-). The resulting boundaries are displayed in the middle of Figure 11.10.

However, this intermediate hull does not consider variances at the subinterval boundaries t_i, e.g., due to the granularity of internal timers and clock drifts. This can be addressed with a symmetric variance in time Δt around the discontinuities t_i. Depending on the function value in the neighboring subinterval in $[-\Delta t, +\Delta t]$ around each discontinuity t_i, either the function value approaching from the left (t_i^-) or the right (t_i^+) is chosen. The resulting acceptance region determined by the bounds $f_i^{(+/-)}$ is displayed on the right of Figure 11.10. A detailed discussion of testing for dynamic power consumption and acceptance region computation is provided in Ref. [50].

11.4.7 Physical Stimulation

Stimuli of physical parameters on the nodes under test can be used to trigger events and to emulate sensor inputs or to exert specific, controlled changes on the nodes. As already discussed before, a

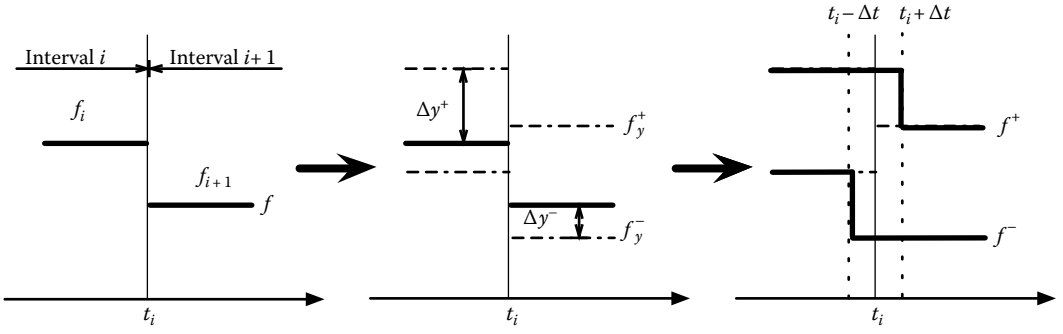


FIGURE 11.10 Illustration of bounds derivation of the acceptance region given a reference function.

controlled RF environment using programmable splitters, multiplexers, and attenuators would also be possible. This is a common technique in the test of RF systems requiring much care to the detail and high investments as the proper shielding of all signals is not trivial. So far the complexity of the implementation has only allowed to use simple triggers based on voltage levels, slopes, and external clock inputs. Nevertheless, any available stimulation to represent deployment conditions increases test coverage, rendering the design more reliable.

11.4.8 Test and Instrumentation Architecture for Physical Characterization

The following section describes our current implementation of the test and instrumentation architecture. The main component is the CruiseControl server, running a number of software projects: TinyOS applications or the PermaSense example discussed in Section 11.5. Our testing architecture is composed of a DSN testbed using a number of different target devices [35] and laboratory instrumentation using an Agilent Multifunction Switch Unit allowing to address multiple sensor node targets for hardware instrumentation, multimeters, precision counter, and a programmable power source/power analyzer (see Figure 11.11). All instruments are accessible over TCP/IP and are

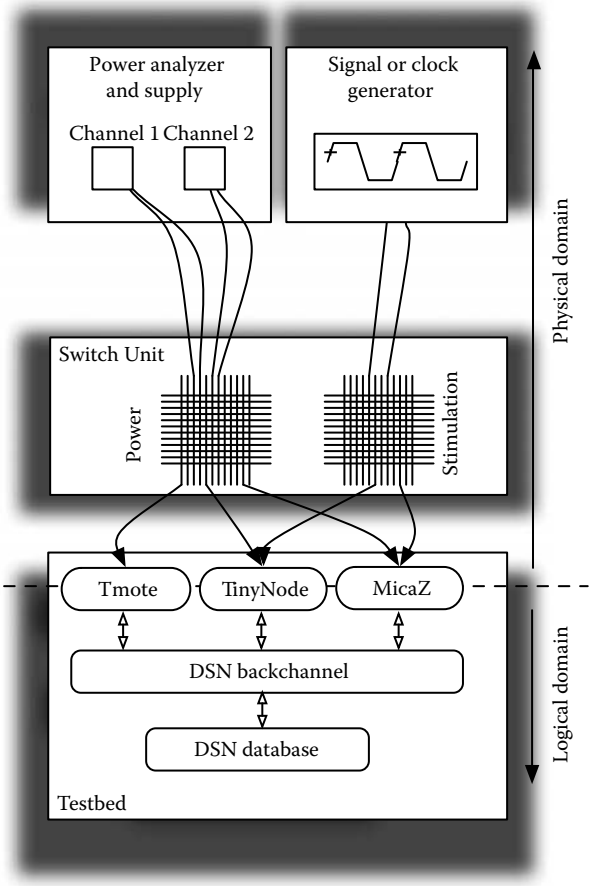


FIGURE 11.11 Logical and physical domain of the test architecture control apparatus used.

FIGURE 11.12 TCT chamber for emulating harsh environmental conditions.

controlled from the CI server using a custom tool. For automated control of the environment, we currently use a prototypical setup of a TCT, built from a refrigerator, an integrated heater, and fans for the airflow, which can be remotely controlled (see Figure 11.12).

11.5 Test and Validation for Life on the Glacier—The PermaSense Case

The PermaSense project [51] aims at deploying a WSN to investigate the processes driving permafrost in steep alpine terrain and as a result of the retreat of the glaciers, also rockfall. The targeted deployment regions in the Swiss alps are above 3500 m a.s.l. and are extremely hostile to any kind of technology, least of all tiny high-tech sensor nodes over a prolonged period of time. Extreme chilling and fast temperature variations, snow and ice, wind, avalanches, and rockfall are properties that result as direct consequences of the area under investigation. Since the processes investigated are only happening very slowly, in some cases on a yearly scale, possibly even multiyear, the systems deployed for measurements must operate over very long periods of time as to minimize the servicing intervals. In a first analysis, logging only-solutions might provide a solution, alleviating the power hungry wireless link. Careful reconsideration of this approach reveals that in a significant amount of locations actually interesting to geoscientists, the avalanches and rockfall prevail and sensor stations might be swept away before a data retrieval by removal of the station could take place. Thus the reliability and power efficiency of a complete WSN-based solution is of primary importance.

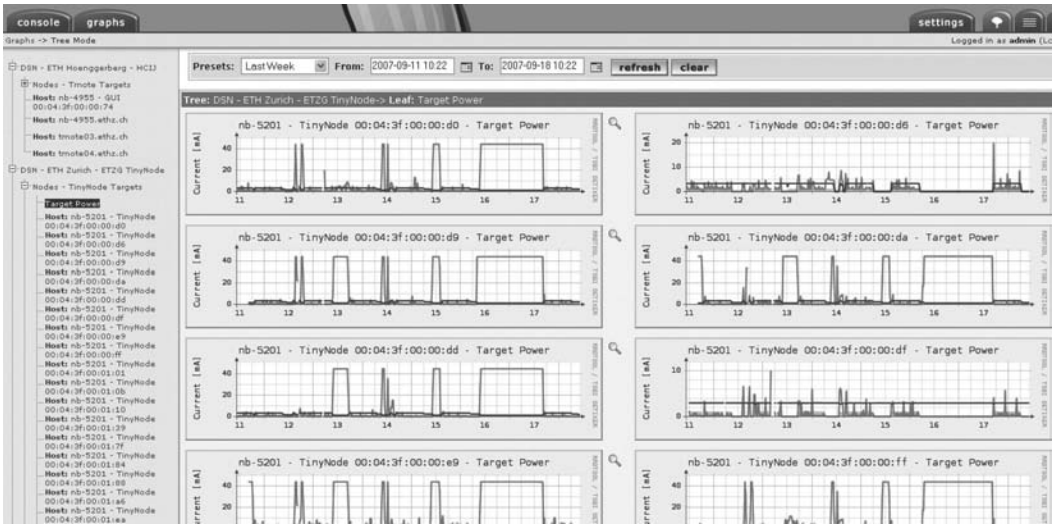
In excess of a detailed specification and careful design phase, for the success of the PermaSense application, rigorous test procedures including temperature cycle testing emulating the condition on the mountain are of primary importance. Due to the fact that traveling to and from the deployment site takes a whole day and can only be performed during the summer a failure of building blocks of the technology would result in a complete failure of the project. In the following we discuss the application of the test and validation methodology developed in the previous sections of this chapter to this application. Apart from describing the steps taken and results obtained we critically discuss our experience gained during this work.

11.5.1 PermaSense System Architecture

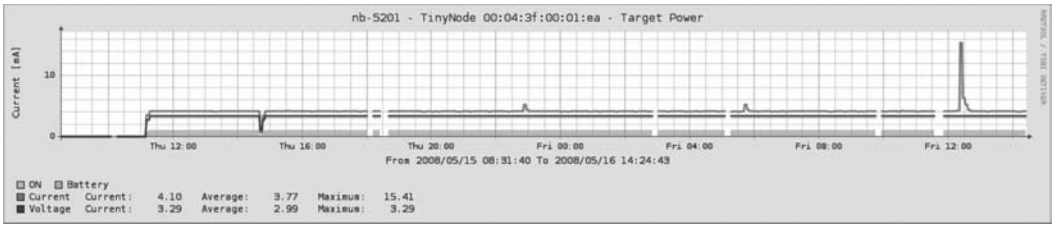
The PermaSense system architecture under tests is based on the Shockfish TinyNode platform [52], a custom data acquisition and power board, a base station containing a satellite modem, specialized Li-$SOCl_2$ batteries, packaged in an IP68-rated protective steel enclosure. The software is based on an application that samples sensors and transmits the data multihop to the base station using a synchronized protocol with acknowledgments. This protocol uses an interleaved scheme of 30 and 5 min for data transmission and synchronization messages, respectively and 5 min synchronization intervals. It has been designed to offer 3 years lifetime based on the batteries used and depending on the influence of the environment and the amount of power used in the dominant communication. Overall the system targets the ambitious goal of a duty cycle of 1.35% and a mean current consumption per node of less than 300 μA.

11.5.2 PermaSense—Testing and Evaluation

After an inaugural development phase and a series of integration builds only using CI (see Figure 11.6) a first number of test jobs for the PermaSense networking and communication component were run on the testbed. This involved basic long-term power trending (see Figure 11.13) and log file


console
graphs
settings
Presets: Last Week
From: 2007-09-11 10:22
To: 2007-09-18 10:22
refresh
clear
nb-5201 - TinyNode 00:04:3f:00:00:d0 - Target Power
nb-5201 - TinyNode 00:04:3f:00:00:d6 - Target Power
nb-5201 - TinyNode 00:04:3f:00:00:d9 - Target Power
nb-5201 - TinyNode 00:04:3f:00:00:da - Target Power
nb-5201 - TinyNode 00:04:3f:00:00:dd - Target Power
nb-5201 - TinyNode 00:04:3f:00:00:df - Target Power
nb-5201 - TinyNode 00:04:3f:00:00:e9 - Target Power
nb-5201 - TinyNode 00:04:3f:00:00:ff - Target Power

(a) Overview of all nodes.



(b) Detailed view for a single TinyNode.

FIGURE 11.13 Long-term trending of power consumption on a DSN testbed. Current and supply voltage are sampled on every node.

analysis. During this early phase of the development this analysis was insightful and in cooperation with a number of detailed project meetings and code reviews lead to an overall improvement of the development process.

Memory profiling on every integration build was integrated into the toolflow as the next step (see Figure 11.14). Subsequently testing became more fine-grained with very specific questions on the roadmap, e.g., the influence of topology variations or excessive clustering of nodes on the networking performance. Sensitivity analysis was performed using specific parametrized versions of the software that was tested in individual test jobs with similar characteristics, e.g., 1 day, fixed number of nodes. For a detailed power/performance analysis, a combined setup of four nodes attached to the instrumentation setup described in Section 11.4.8 with two nodes inside the TCT chamber and two nodes outside as well as one node additionally attached to a precision counter for monitoring the stability of the system clock over longer periods of time was used. A basic current and voltage profile with the characteristics of the time division multiple access (TDMA) communication scheme of the PermaSense protocol can be seen in Figure 11.15. It can be seen that basically the protocol performs as expected with the four nodes in sync. Only occasionally a node in the TCT chamber needs to resync (longer active phase seen in period 3, 9, and 13) due to the influence of the cooling. A detailed view on one of the communication rounds is shown in Figure 11.15. Here the effect of high power load on an already drained battery can be seen when load increases in the active phase of the protocol. With a lower operating limit of 2.4 V as given for the TinyNode, it is clear that we are already reaching a case of possible malfunction of the system here although the battery seems to be still healthy when the node is sleeping. Taking a closer look at the current consumption in sleep mode by zooming in on the current traces taken (see Figure 11.16) reveals considerable differences on the four nodes under test. Clearly there are device variations and spurious errors visible, but it seems as if there are also systematic errors (the lowest curve shows repetitive, unstable behavior). Even worse, this (mis-) behavior has been observed on different nodes, but it could not be observed on all nodes! To date we have not been able to find out all details of what is happening here, but the analysis has sharpened our

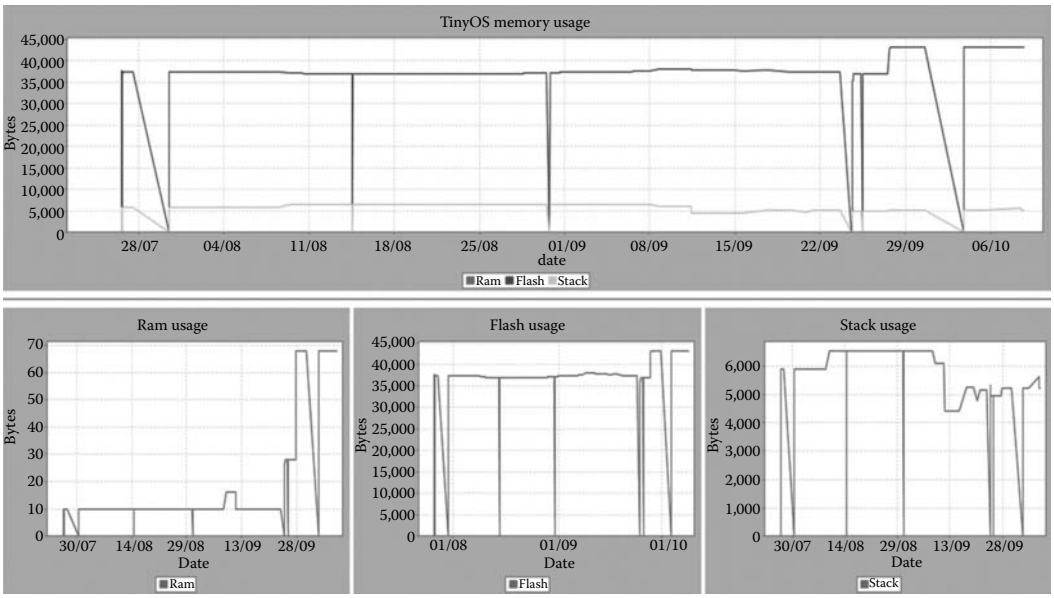


FIGURE 11.14 Evolution of memory usage over three months of the development process.

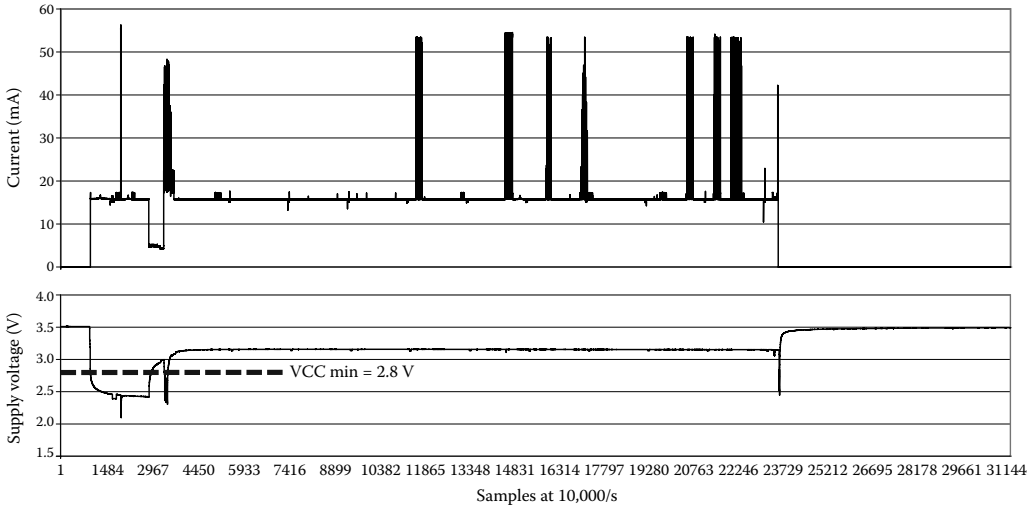


FIGURE 11.15 Current trace for one communication round at 10,000 samples/s.

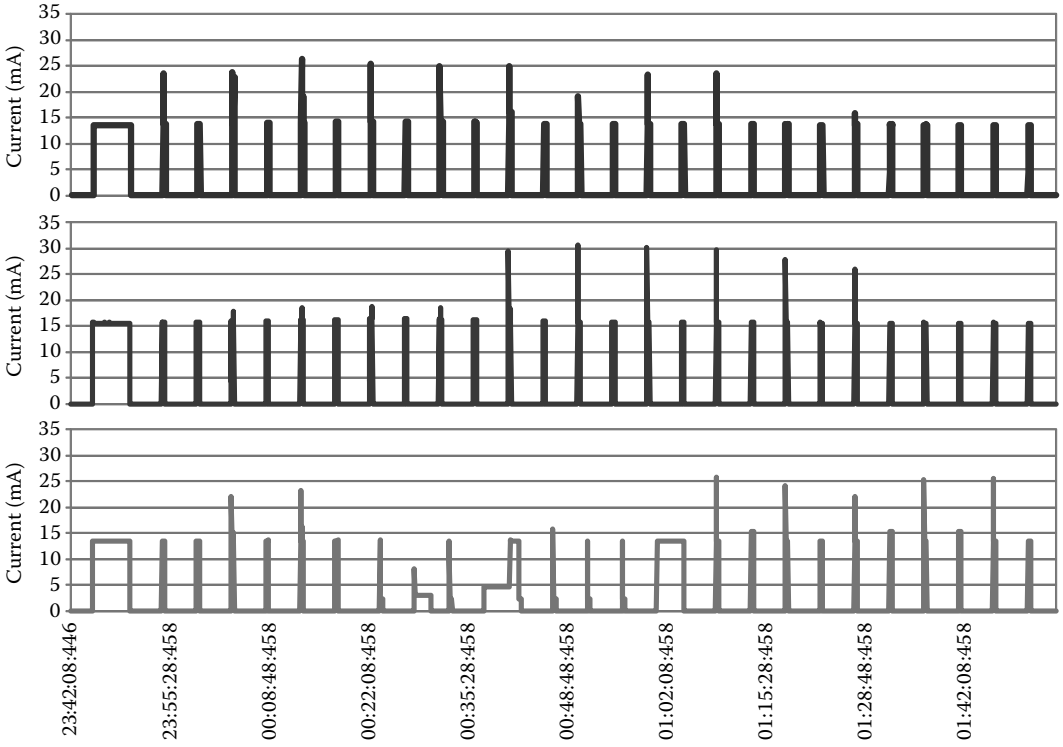


FIGURE 11.16 Current consumption of three independent nodes running a TDM protocol. The bottom node can be seen losing sync and re synchronizing after a number of iterations.

mind and clearly requires attention to the effects visible in greater detail. However in the context of this chapter we refrain from going into the full (numerical) details of the analysis of the PermaSense application as the objective here is to highlight the methodology, procedures, and tool integration developed for a systematic analysis of WSN applications.

11.6 Summary

WSNs are distributed embedded systems communicating wirelessly, typically via radio. The combination of the embedded devices tightly integrated into the environment with unreliable communication over the wireless channel and longevity design goals renders WSN system design and development a challenging task. Validation and testing methodologies and tools need to provide support to arrive at a functionally correct, robust, and long-living system at deployment time. Different tools need to be used to allow for addressing the different characteristics of the system; the presented integration of tools into a comprehensive test architecture allows for leveraging individual benefits in a CI framework.

References

1. J. M. Kahn, R. H. Katz, and K. S. J. Pister, Next century challenges: Mobile networking for smart dust, in *Proc. 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*. ACM Press, New York, 1999, pp. 271–278.
2. S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, Coverage problems in wireless ad-hoc sensor networks, in *INFOCOM*, Anchorage, AK, 2001, pp. 1380–1387.
3. P. Dutta, J. Hui, J. Jeong, S. Kim, C. Sharp, J. Taneja, G. Tolle, K. Whitehouse, and D. Culler, Trio: Enabling sustainable and scalable outdoor wireless sensor network deployments, in *Proc. 5th Int'l Conf. Information Processing Sensor Networks (IPSN '06)*, Nashville, TN, 2006, pp. 407–415.
4. D. Kotz, C. Newport, and C. Elliott, The mistaken axioms of wireless-network research, Dartmouth College Computer Science, Tech. Rep. TR2003–467, Hanover, NH, 2003.
5. K. Römer and F. Mattern, The design space of wireless sensor networks, *IEEE Wireless Communications*, 11(6), 54–61, December 2004.
6. R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, Lessons from a sensor network expedition, in *Proc. 1st European Workshop on Sensor Networks (EWSN 2004)*, Berlin, Germany, 2004, pp. 307–322.
7. J. Choi, J. Lee, M. Wachs, and P. Levis, Opening the sensornet black box, *SIGBED Rev.*, 4(3), 13–18, 2007. doi=http://doi.acm.org/10.1145/1317103.1317106
8. K. Langendoen, A. Baggio, and O. Visser, Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture, in *Proc. 20th Int'l Parallel and Distributed Processing Symposium (IPDPS 2006)*, Rhodes Island, Greece, 2006, pp. 8–15.
9. P. K. Haneveld, Evading murphy: A sensor network deployment in precision agriculture, June 2007. [Online]. Available at: http://www.st.ewi.tudelft.nl/~koen/papers/LOFAR-agro-take2.pdf
10. S. Rost and H. Balakrishnan, Memento: A health monitoring system for wireless sensor networks, in *Proc. 3rd IEEE Communications Society Conf. Sensor, Mesh and Ad Hoc Communications and Networks (IEEE SECON 2006)*, Reston, VA, 2006, 574–584.
11. M. Wachs, J. I. Choi, J. W. Lee, K. Srinivasan, Z. Chen, M. Jain, and P. Levis, Visibility: A new metric for protocol design, in *Proc. 5th ACM Conf. Embedded Networked Sensor Systems (SenSys 2007)*, 2007, Sydney, Australia, pp. 73–86.
12. M. Ringwald, K. Römer, and A. Vitaletti, Passive inspection of sensor networks, in *Proc. 3rd IEEE Int'l Conf. Distributed Computing in Sensor Systems (DCOSS 2007)*, Santa Fe, NM, June 2007.
13. K. Whitehouse, G. Tolle, J. Taneja, C. Sharp, S. Kim, J. Jeong, J. Hui, P. Dutta, and D. Culler, Marionette: Using RPC for interactive development and debugging of wireless embedded networks, in *Proc. 5th Int'l Conf. Information Processing Sensor Networks (IPSN '06)*. ACM Press, New York, 2006, pp. 416–423.
14. H. Liu, L. Selavo, and J. Stankovic, Seedtv: Deployment-time validation for wireless sensor networks, in *Proc. 4th IEEE Workshop on Embedded Networked Sensors (EmNetS-IV)*, Cork, Ireland, 2007, pp. 23–27.

15. J. Yang, M. L. Soffa, L. Selavo, and K. Whitehouse, Clairvoyant: A comprehensive source-level debugger for wireless sensor networks, in *Proc. 5th ACM Conf. Embedded Networked Sensor Systems (SenSys 2007)*, Sydney, Australia, 2007, 189–203.
16. P. Corke, P. Valencia, P. Sikka, T. Wark, and L. Overs, Long-duration solar-powered wireless sensor networks, in *Proc. 4th IEEE Workshop on Embedded Networked Sensors (EmNetS-IV)*, Cork, Ireland, 2007, pp. 33–37.
17. J. Rabaey, M. Ammer, J. da Silva Jr., D. Patel, and S. Roundy, PicoRadio supports ad hoc ultra-low power wireless networking, *IEEE Computer*, 33(7), 42–48, July 2000.
18. G. Pottie and W. Kaiser, Wireless integrated network sensors, *Communications of the ACM*, 43(5), 51–58, May 2000.
19. M. Woehrle, C. Plessl, R. Lim, J. Beutel, and L. Thiele, EvAnT: Analysis and checking of event traces for wireless sensor networks, in *Proc. 2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC 2008)*, Washington, DC, June 2008, pp. 201–208.
20. R. Schwarz and F. Mattern, Detecting causal relationships in distributed computations: In search of the holy grail, *Distributed Computing*, 7(3), 149–174, 1994. [Online]. Available at: citeseer.ist.psu.edu/article/schwarz94detecting.html
21. R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, Habitat monitoring with sensor networks, *Commun. ACM*, 47(6), 34–40, 2004.
22. Texas Instruments, Msp430 ultra-low power microcontrollers overview from Texas Instruments, September 2007. [Online]. Available at: http: //focus.ti.com/mcu/docs/mcuhome.tsp?sectionId=101
23. J. P. Bowen and M. G. Hinchey, Ten commandments of formal methods . . .ten years later, *Computer*, 39(1), 40–48, 2006.
24. D. E. Michael Allen and L. Girod, Acoustic laptops as a research enabler, in *Proc. 4th IEEE Workshop on Embedded Networked Sensors (EmNetS-IV)*, Cork, Ireland, 2007, pp. 63–67.
25. R. King and L. Atallah, An hmm framework for optimal sensor selection with applications to bsn sensor glove design, in *Proc. 4th IEEE Workshop on Embedded Networked Sensors (EmNetS-IV)*, Cork, Ireland, 2007, pp. 58–62.
26. Sun, WSN platform, http://www.sunspotworld.com/.
27. J. Gray, Why do computers stop and what can be done about it? in *Proc. 5th Symp. Reliability in Distributed Software and Database Systems (SRDS 86)*, Los Angeles, CA, January 1986, pp. 3–12.
28. P. Levis, N. Lee, M. Welsh, and D. Culler, TOSSIM: Accurate and scalable simulation of entire TinyOS applications, in *Proc. 1st ACM Conf. Embedded Networked Sensor Systems (SenSys 2003)*, Los Angeles, CA, November 2003, pp. 126–137.
29. M. Maifi, H. Khan, T. Abdelzaher, and K. K. Gupta, Towards diagnostic simulation in sensor networks, in *Distributed Computing in Sensor Systems*. Springer, Berlin/Heidelberg, 2008.
30. GloMoSim, Network simulator, http://pcl.cs.ucla.edu/projects/glomosim
31. National ICT Australia, National ICT Australia—Eastalia [home], September 2007. [Online]. Available at: http://castalia.npc.nicta.com.au/
32. Main page Nsnam, October 2007. [Online]. Available at: http://nsnam.isi.edu/nsnam/index.php/
33. L. Girod, N. Ramanathan, J. Elson, T. Stathopoulos, M. Lukac, and D. Estrin, Emstar: A software environment for developing and deploying heterogeneous sensor–actuator networks, *ACM Transactions on Sensor Networks*, 3(3), 13, 2007.
34. G. Werner-Allen, P. Swieskowski, and M. Welsh, MoteLab: A wireless sensor network testbed, in *Proc. 4th Int'l Conf. Information Processing Sensor Networks (IPSN '05)*, April 2005, pp. 483–488.
35. M. Dyer, J. Beutel, L. Thiele, T. Kalt, P. Oehen, K. Martin, and P. Blum, Deployment support network—A toolkit for the development of WSNs, in *Proc. 4th European Workshop on Sensor Networks (EWSN 2007)*, Delft, the Netherlands, 2007, pp. 195–211.
36. V. Turau, M. Witt, and M. Venzke, Field trials with wireless sensor networks: Issues and remedies, in *Proc. of the Int'l Multi-Conference on Computing in the Global Information Technology (ICCGI '06)*, 2006, p. 86.

37. F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, Cross-level sensor network simulation with COOJA, in *Local Computer Networks, Proc. 2006 31st IEEE Conference on*, Tampa, FL, November 2006, pp. 641–648.
38. M. Woehrle, C. Plessl, J. Beutel, and L. Thiele, Increasing the reliability of wireless sensor networks with a distributed testing framework, in *Proc. 4th IEEE Workshop on Embedded Networked Sensors (EmNetS-IV)*. ACM, New York, 2007, pp. 93–97.
39. B. Meyer, Design by contract: Making object-oriented programs that work, in *Technology of Object-Oriented Languages and Systems, 1997. TOOLS 24, Proceedings*, Beijing, China, November 1997, pp. 360–361.
40. W. Archer, P. Levis, and J. Regehr, Interface contracts for tinyos, in *Proc. 6th Int'l Conf. Information Processing Sensor Networks (IPSN '07)*. ACM Press, New York, 2007, pp. 158–165.
41. A. Lachenmann, ncunit: A unit testing framework for nesc, 2008. [Online]. Available at: http://www.ipvs.uni-stuttgart.de/abteilungen/vs/abteilung/mitarbeiter/eigenes/lachenas/ncunit/start
42. Rincon Research Corporation, TinyOS 2.x automated unit testing/tunit, http://www.lavalampmotemasters.com/, May 2008.
43. N. T. M. Nguyen and M. L. Soffa, Program representations for testing wireless sensor network applications, in *DOSTA '07: Workshop on Domain Specific Approaches to Software Test Automation*. ACM, Dubrovnik, Croatia, 2007, pp. 20–26.
44. V. Handziski, A. Koepke, A. Willig, and A. Wolisz, Twist: A scalable and reconfigurable testbed for wireless indoor experiments with sensor networks, in *Proc. 2nd International Workshop on Multi-Hop Ad Hoc Networks: From Theory to Reality (REALMAN '06)*. ACM Press, New York, 2006, pp. 63–70.
45. A. G. Paul Duvall and S. Matyas, *Continuous Integration: Improving Software Quality and Reducing Risk*. Addison-Wesley, Boston, MA, 2007.
46. D. Cohen, M. Lindvall, and P. Costa, An introduction to agile methods. *Advances in Computers*, 62, 2–67, 2004.
47. CruiseControl, Cruisecontrol home, April 2008. [Online]. Available at: http://cruisecontrol.sourceforge.net/
48. V. Massol and T. Husted, *JUnit in Action*. Manning Publications Co., Greenwich, CT, 2003.
49. X. Jiang, P. Dutta, D. Culler, and I. Stoica, Micro power meter for energy monitoring of wireless sensor networks at scale, in *IPSN '07: Proc. 6th International Conference on Information Processing in Sensor Networks*. ACM, New York, 2007, pp. 186–195.
50. M. Woehrle, J. Beutel, R. Lim, M. Yuecel, and L. Thiele, Power monitoring and testing in wireless sensor network development, in *Workshop on Energy in Wireless Sensor Networks (WEWSN 2008)*, June 2008.
51. I. Talzi, A. Hasler, S. Gruber, and C. Tschudin, Permasense: Investigating permafrost with a WSN in the Swiss Alps, in *Proc. 4th IEEE Workshop on Embedded Networked Sensors (EmNetS-IV)*, ACM Press, New York, 2007, 8–12.
52. H. Dubois-Ferriere, R. Meier, L. Fabre, and P. Metrailler, Tinynode: a comprehensive platform for wireless sensor network applications, in *Proc. 5th Int'l Conf. Information Processing Sensor Networks (IPSN '06)*, Boulder, CO, April 2006, pp. 358–365.

12

Developing and Testing of Software for Wireless Sensor Networks

Jan Blumenthal
SYSGO AG

Frank Golatowski
University of Rostock

Ralf Behnke
University of Rostock

Steffen Prüter
University of Rostock

Dirk Timmermann
University of Rostock

12.1 Introduction **12**-1
12.2 Preliminaries **12**-2
Architectural Layer Model • Middleware and Services for Sensor Networks • Programming Aspect vs. Behavioral Aspect
12.3 Software Architectures **12**-5
TinyOS • MATÉ • TinyDB • SensorWare • MiLAN • EnviroTrack • SeNeTs • Contiki • Sensor Web Enablement
12.4 Simulation, Emulation, and Test of Large-Scale Sensor Networks **12**-22
A TinyOS SIMulator • EmStar • SeNeTs—Test and Validation Environment • J-Sim
12.5 Mastering Deployed Sensor Networks **12**-31
12.6 Summary **12**-34
References **12**-34

12.1 Introduction

The increasing miniaturization of electronic components and advances in modern communication technologies enable the development of high-performance spontaneously networked and mobile systems. Wireless microsensor networks promise novel applications in several domains. Forest fire detection, battlefield surveillance, or telemonitoring of human physiological data are only in the vanguard of plenty of improvements encouraged by the deployment of microsensor networks. Hundreds or thousands of collaborating sensor nodes form a microsensor network. Sensor data is collected from the observed area, locally processed or aggregated, and transmitted to one ore more base stations.

Sensor nodes can be spread out in dangerous or remote environments whereby new application fields can be opened. A sensor node combines the abilities to compute, communicate, and sense. Figure 12.1 shows the structure of a typical sensor node consisting of a processing unit, a communication module (radio interface), and sensing and actuator devices.

Figure 12.2 shows a scenario taken from the environmental application domain: leakage detection of dykes. During floods, sandbags are used to reinforce dykes. Piled along hundreds of kilometers around lakes or rivers, sandbag dykes keep waters at bay and bring relief to residents. Sandbags are stacked against sluice gates and parts of broken dams to block off the tide. To find out spots of leakage each sandbag is equipped with a moisture sensor and transmits sensor data to a base station next to

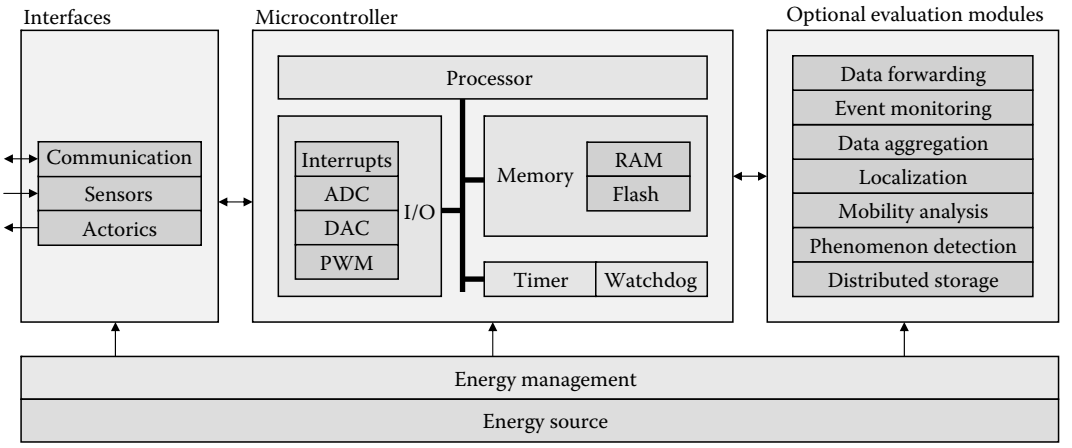


FIGURE 12.1 Structure of a sensor node.

the dyke. Thus, leakages can be detected earlier and reinforcement actions can be coordinated more efficiently.

Well-known research activities in the field of sensor networks are UCLA's WINS [3], Berkeley's Smart Dust [1], WEBS [2], and PicoRadio [4]. An example of European research activities is the EYES-Project [5]. Detailed surveys on sensor networks can be found in Refs. [6,7]. This chapter focuses on innovative architectures and basic concepts of current software development solutions for wireless sensor networks.

12.2 Preliminaries

Central unit of a sensor node is a low-power microcontroller that controls all functional parts. Software for such a microcontroller has to be resource aware on one hand. On the other hand, several quality-of-service (QoS) aspects have to be met by sensor node software, such as latency, processing time for data fusion or compression, or flexibility regarding routing algorithms or medium access control (MAC) techniques.

Conventional software development for microcontrollers usually covers hardware abstraction layer (HAL), operating system (OS) and protocols, and application layer. Often, software for microcontrollers is limited to an application-specific monolithic software block that is optimized for

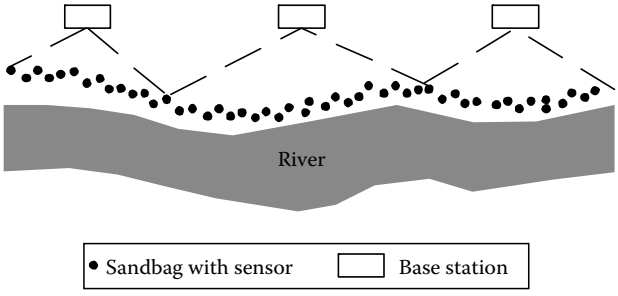


FIGURE 12.2 Example sensor network application: leakage detection along dikes.

performance and resource usage. Abstracting layers, such as HAL or OS, are often omitted due to resource constraints and low-power aspects.

Microcontrollers are often developed and programmed for a specific, well-defined task. This limitation of the application domain leads to high-performance embedded systems even with strict resource constraints. Development and programming of such systems are too much effort. Furthermore, an application developed for one microcontroller is in most cases not portable to any other one, so that it has to be reimplemented from scratch. Microcontroller and application form an inseparable unit. If the application domain of an embedded system changes often, the whole microcontroller must be replaced instead of writing and downloading a new program.

For sensor nodes, application-specific microcontrollers are preferred instead of general-purpose microprocessors. This is because of the small size and the low energy consumption of those controllers. However, requirements concerning a sensor node exceed the main characteristics of a conventional microcontroller and its software. The main reason for this is the dynamic character of a sensor node's task. Sensor nodes can adopt different tasks, such as sensor data acquisition, data forwarding, or information processing. The task assigned to a node with its deployment is not fixed until the end of its life cycle. Depending on, for instance location, energy level, or neighborhood of a sensor node, a task change can become advantageous or even necessary.

Additionally, software for sensor nodes should be reusable. An application running on a certain sensor node should not be tied to a specific microcontroller but to some extent be portable onto different platforms to enhance interoperability of sensor nodes with different hardware platforms. Not limited to software development for wireless sensor networks is the general requirement for a straightforward programmability and, as a consequence, a short development time.

It is quite hard or even impossible to meet the requirements mentioned above with a monolithic application. Hence, at present there is much research effort in the areas of middleware and service architectures for wireless sensor networks. A middleware for wireless sensor networks should encapsulate required functionality in a layer between OS and application. Incorporating a middleware layer has the advantage that applications get smaller and are not tied to a specific microcontroller. At the same time, development effort for sensor node applications (SNA) reduces since a significant part of the functionality moves from application to middleware. Another research domain tends to service architectures for wireless sensor networks. A service layer is based on mechanisms of a middleware layer and makes its functionality more usable.

12.2.1 Architectural Layer Model

Like in other networking systems, the architecture of a sensor network can be divided into different layers (see Figure 12.3). The lower layers are the hardware and HAL. The OS layer and protocols are above the hardware-related layers. The OS provides basic primitives, such as multithreading, resource management, and resource allocation that are needed by higher layers. Also access to radio interface and input/output operations to sensing devices are supported by basic OS primitives. Usually in node-level OSs these primitives are rudimentary and there is no separation between user and kernel mode. On top of the OS layer reside middleware, service, and application layer.

In recent years, much work has been done to develop sensor network node devices (e.g., Berkeley Motes [9]), OSs, and algorithms, for example, for location awareness, power reduction, data aggregation, and routing. Today researchers are working on extended software solutions including middleware and service issues for sensor networks. The main focus of these activities is to simplify application development process and to support dynamic programming of sensor networks. The overall development process of sensor node software usually ends with a manual download of an executable image over direct wired connections or over-the-air (OTA) interface to the target node.

After deployment of nodes, it is nearly impossible to improve or adapt new programs to the target nodes. But this feature is necessary in future wireless sensor networks to adapt the behavior of

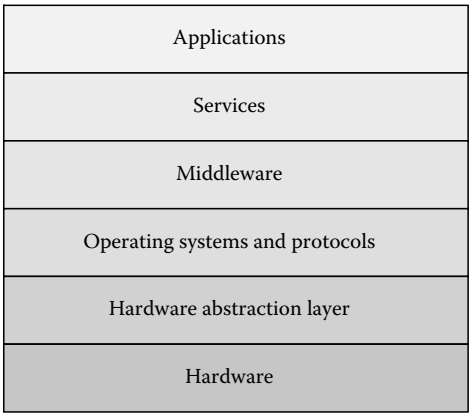


FIGURE 12.3 Layered software model.

sensor network dynamically through new injected programs or capsules. The task assigned to a node with its deployment is not fixed until the end of its life cycle. Depending on, for instance, location, energy level, or neighborhood of a sensor node, a task change can become advantageous or even necessary.

12.2.2 Middleware and Services for Sensor Networks

In sensor networks, design and development of solutions for higher-level middleware functionality and creation of service architectures are an open research issue. Middleware for sensor networks has two primary goals:

- Support of acceptable middleware application programming interfaces (APIs), which abstract and simplify low-level APIs to ease application software development and to increase portability
- Distributed resource management and allocation

Besides the native network functions, such as routing and packet forwarding, future software architectures are required to enable location and utilization of services. A service is a program that can be accessed through standardized functions over a network. Services allow a cascading without previous knowledge of each other, and thus enable the solution of complex tasks. A typical service used during the initialization of a node is the localization of a data sink for sensor data. Gateways or neighboring nodes can provide this service. To find services, nodes use a service discovery protocol.

12.2.3 Programming Aspect vs. Behavioral Aspect

Wireless sensor networks do not have to consist of homogeneous nodes. In reality, a network composed of several groups of different sensor nodes is imaginable. This fact changes the software development approach and points out new challenges as they are well known from the distributed systems domain. In an inhomogeneous wireless sensor network, nodes contain different low-level system APIs however with similar functions. From a developer's point of view it is hard to create programs, since APIs are mostly incompatible. To overcome the mentioned problems in heterogeneity and complexity, new software programming techniques are required. One attempt to accomplish this aspect is the definition of an additional API or an additional class library on top of each system API. But they are all limited by some means or other, for example, platform independency, flexibility,

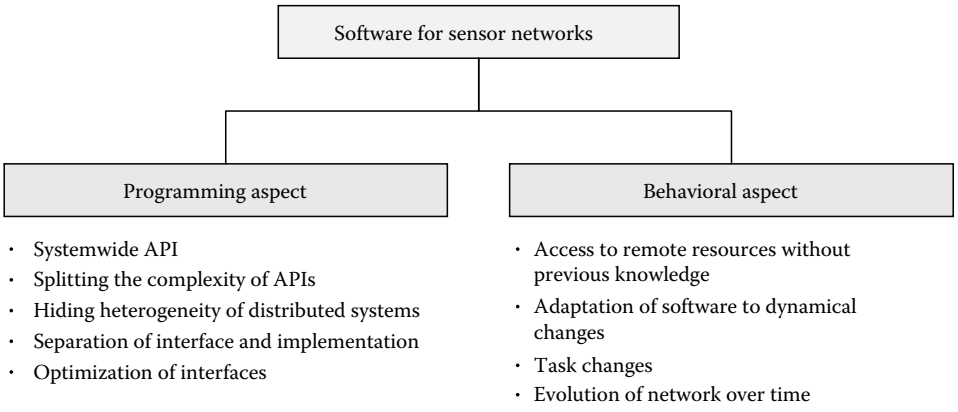

Software for sensor networks
Programming aspect
Behavioral aspect
• Systemwide API
• Splitting the complexity of APIs
• Hiding heterogeneity of distributed systems
• Separation of interface and implementation
• Optimization of interfaces
• Access to remote resources without previous knowledge
• Adaptation of software to dynamical changes
• Task changes
• Evolution of network over time

FIGURE 12.4 Two aspects of software for wireless sensor networks.

quantity, and programming language. All approaches to achieve an identical API on different systems are covered by the programming aspect (Figure 12.4).

The programming aspect enables the developer to easily create programs on different hardware and software platforms. But an identical API on all platforms does not necessarily take the dynamics of the distributed system into account. Ideally, the application does not notice any dynamic system changes. This decoupling is termed as *behavioral aspect* and covers:

- Access to remote resources without previous knowledge, for example, remote procedure calls (RPCs) and discovered services
- Adaptations within the middleware layer to dynamic changes in the behavior of a distributed system, caused by incoming or leaving resources, mobility of nodes, or changes of the environment
- Ability of the network to evolve over time including modifications of the system's task, exchange or adaptation of running software parts, and mobile agents

12.3 Software Architectures

This chapter presents five important software solutions for sensor networks. It starts with the most mature development TinyOS and depending software packages. It continues with SensorWare followed by two promising concepts, middleware linking applications and networks (MiLAN) and EnviroTrack. The section finalizes with an introduction to SeNeTs that features interface optimization.

12.3.1 TinyOS

TinyOS is a component-based OS for sensor networks developed at UC Berkeley. TinyOS can be seen as an advanced software framework [9] that has a large user community due to its open-source character and its promising design. The framework contains numerous prebuilt sensor applications and algorithms, for example, multihop ad hoc routing and supports different sensor node platforms. Originally it was developed for Berkeley's Mica Motes. Programmers experienced with the C programming language can easily develop TinyOS applications written in a proprietary language called NesC [10].

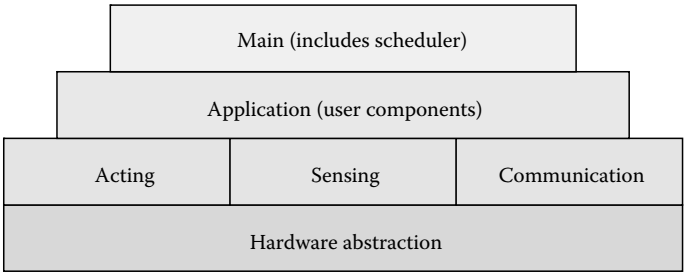


FIGURE 12.5 Software architecture of TinyOS.

The design of TinyOS is based on the specific sensor network characteristics: small physical size, low-power consumption, concurrency-intensive operation, multiple flows, limited physical parallelism and controller hierarchy, diversity in design and usage, and robust operation to facilitate the development of reliable distributed applications. The main intention of the TinyOS developers was "retaining energy, computational and storage constraints of sensor nodes by managing the hardware capabilities effectively, while supporting concurrency-intensive operation in a manner that achieves efficient modularity and robustness" [16]. Therefore, TinyOS is optimized in terms of memory usage and energy efficiency. It provides defined interfaces between the components that reside in neighboring layers. A layered model is shown in Figure 12.5.

12.3.1.1 Elemental Properties

TinyOS utilizes an event model instead of a stack-based threaded approach, which would require more stack space and multitasking support for context switching, to handle high levels of concurrency in a very small amount of memory space. Event-based approaches are the favorite solution to achieve high performance in concurrency intensive applications. Additionally, the event-based approach uses CPU resources more efficiently and therefore takes care of the most precious resource, the energy.

An event is serviced by an event handler. More complex event handling can be done by a task. The event handler is responsible for posting the task to the task scheduler. Event and task scheduling is performed by a two-level scheduling structure. This kind of scheduling provides that events, associated with a small amount of processing, can be performed immediately, while longer running tasks can be interrupted by events. Tasks are handled rapidly, however, no blocking or polling is permitted.

The TinyOS system is designed to scale with the technology trends supporting both, smaller designs and crossover of software components into hardware. The latter provides a straightforward integration of software components into hardware.

12.3.1.2 TinyOS Design

The architecture of a TinyOS system configuration is shown in Figure 12.6. It consists of the tiny scheduler and a graph of components. Components satisfy the demand for modular software architectures. Every component consists of four interrelated parts: a command handler, an event handler, an encapsulated fixed-size and statically allocated frame, and a bundle of simple tasks. The frame represents the internal state of the component. Tasks, commands, and handlers execute in the context of the frame and operate on its state. In addition, the component declares the commands it uses and the events it signals. Through this declaration, modular component graphs can be composed. The composition process creates layers of components. Higher-layer components issue commands to lower-level components and these signal events to higher-level components. To provide an abstract

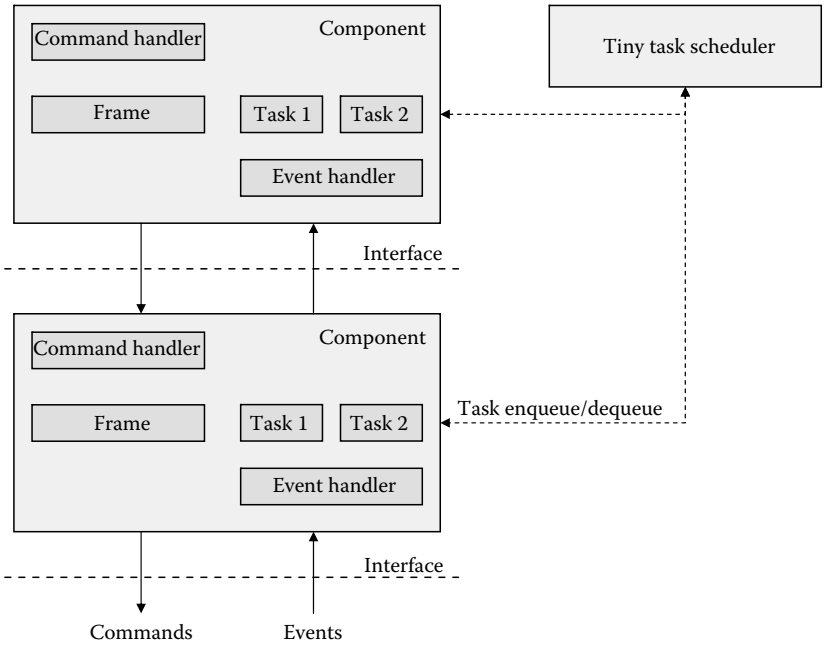


FIGURE 12.6 TinyOS architecture in detail.

definition of the interaction of two components via commands and events, the bidirectional interface is introduced in TinyOS.

Commands are nonblocking requests made to lower-layer components. A command provides feedback to its caller by returning status information. Typically, the command handler puts the command parameters into the frame and posts a task into the task queue for execution. The acknowledgment whether the command was successful, can be signaled by an event.

Event handlers are invoked by events of lower-layer components, or when directly connected to the hardware, by interrupts. Similar to commands, the frame will be modified and tasks are posted. Both, commands and tasks, perform a small fixed amount of work similar to interrupt service routines.

Tasks perform the primary work. They are atomic, run to completion, and can only be preempted by events. Tasks are queued in a first in first out (FIFO) task scheduler to perform an immediate return of event or command handling routines. Due to the FIFO scheduling, tasks are executed sequentially and should be short. Alternatively to the FIFO task scheduler, priority-based or deadline-based schedulers can be implemented into the TinyOS framework.

TinyOS distinguishes three categories of components. Hardware abstraction components map physical hardware into the component model. Mostly, this components export commands to the underlying hardware and handle hardware interrupts. Synthetic hardware components extend the functionality of hardware abstraction components by simulating the behavior of advanced hardware functions, for example, bit-to-byte transformation functions. For future hardware releases, these components can directly cast into hardware. High-level software components perform application-specific tasks, for example, control, routing, data transmission, calculation on data, and data aggregation.

An interesting aspect of the TinyOS framework is the similarity of the component description to the description of hardware modules in hardware description languages, e.g., VHSIC hardware description language (VHDL) or Verilog. A hardware module, for example, in VHDL is defined by an entity with input and output declarations, status registers to hold the internal state, and a finite

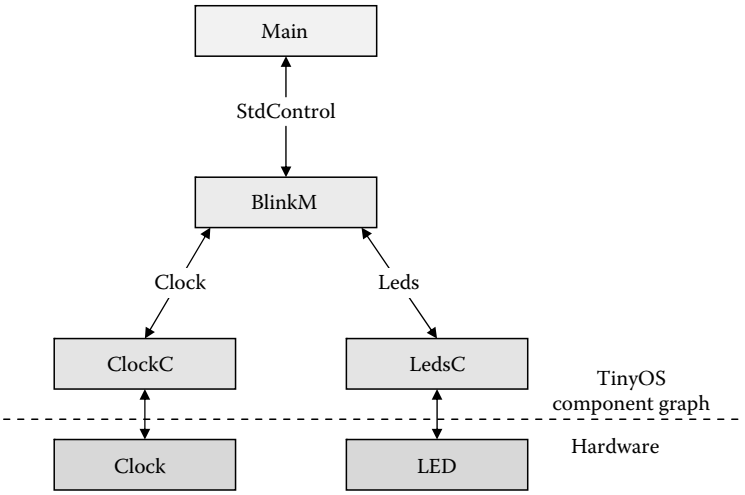


FIGURE 12.7 Simple TinyOS application.

state machine controlling the behavior of the module. In comparison, a TinyOS component contains commands and events, a frame and a behavioral description. These similarities simplify the cast of TinyOS components to hardware modules. Future sensor node generations can benefit from this similarity in describing hardware and software components.

12.3.1.3 TinyOS Application

A TinyOS application consists of one or more components. These components are separated into modules and configurations. Modules implement application-specific code, whereas configurations wire different components together. By using a top-level configuration, wired components can be compiled and linked to form an executable. The interfaces between the components declare a set of commands and events which provide an abstract description of components. The application developer has to implement the appropriate handling routine into the component.

Figure 12.7 shows the component graph of a simple TinyOS application, that turns an LED on and off depending on the clock. The top-level configuration contains the application-specific components (ClockC, LedsC, and BlinkM) and a OS-specific component providing the tiny task-scheduler and initialization functions. The main component encapsulates the TinyOS-specific components from the application. StdControl, Clock, and Leds are the interfaces used in this application. While BlinkM contains the application code, ClockC and LedsC are again configurations encapsulating further component graphs controlling the hardware clock and the LEDs connected to the controller. TinyOS provides a variety of additional extensions, such as the virtual machine (VM) MATÉ and the database TinyDB for cooperative data acquisition.

12.3.2 MATÉ

MATÉ [8] is a byte-code interpreter for TinyOS. It is a tiny communication-centric VM designed as a component for the system architecture of TinyOS. MATÉ is located in the component graph on top of several system components, represented by sensor components, network component, timer component, and nonvolatile storage component.

The developer motivation for MATÉ was to solve novel problems in sensor network management and programming, in response to changing tasks, for example, exchange of the data aggregation

function or routing algorithm. However, the associated inevitable reprogramming of hundreds or thousands of nodes is restricted to energy and storage resources of sensor nodes. Furthermore, the network is limited in bandwidth and network activity as a large energy draw. MATÉ attempts to overcome these problems, by propagating so-called code capsules through the sensor network. The MATÉ VM provides the possibility to compose a wide range of sensor network applications by the use of a small set of higher-level primitives. In MATÉ, these primitives are one-byte instructions and they are stored into capsules of 24 instructions together with identifying and versioning information.

12.3.2.1 MATÉ Architecture

MATÉ is a stack-based architecture that allows a concise instruction set. The use of instructions hides the asynchronous character of native TinyOS programming; because instructions are executed successively as several TinyOS tasks.

The MATÉ VM shown in Figure 12.8 has three execution contexts: Clock, Send, and Receive, which can run concurrently at instruction granularity. Clock corresponds to timer events and Receive to message receive events, signaled from the underlying TinyOS components. Send can only be invoked from the Clock or Receive context. Each context holds an operand stack for handling data and a return stack for subroutines calls. Subroutines allow programs to be more complex as a single capsule can provide. Therefore, MATÉ has four spaces for subroutine code.

The code for the contexts and the subroutines is installed dynamically at runtime by code capsules. One capsule fits into the code space of a context or subroutine. The capsule installation process

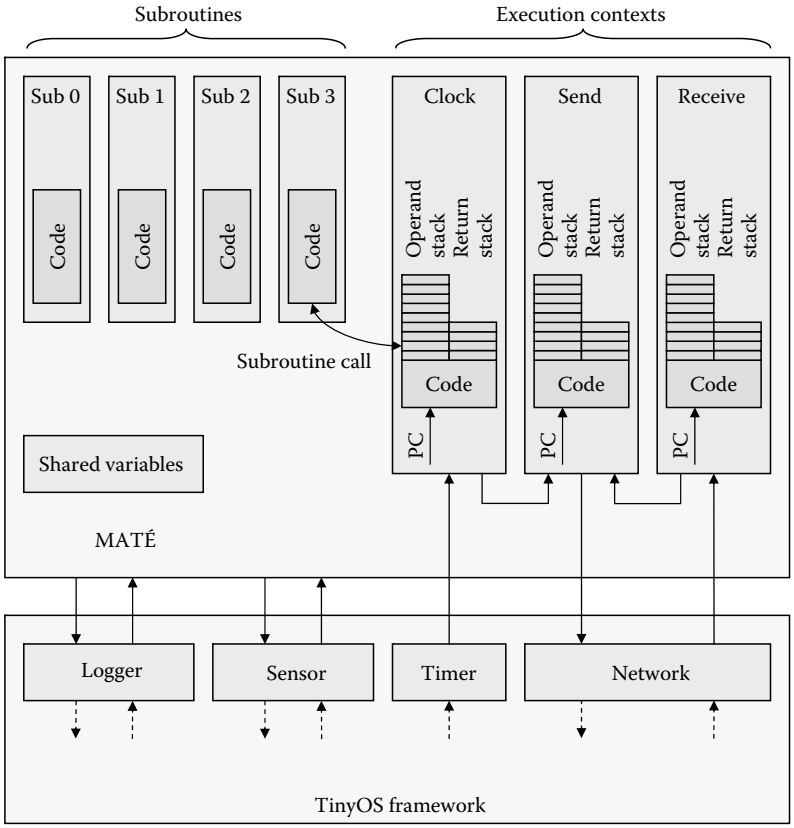


FIGURE 12.8 MATÉ architecture.

supports self-forwarding of capsules to reprogram a whole sensor network with new capsules. It is the task of the sensor network operator to inject code capsules in order to change the behavior of the network.

The program execution in MATÉ starts with a timer event or a packet receive event. The program counter jumps to the first instruction of the corresponding context (Clock or Receive) and executes until it reaches the Halt instruction. Each context can call subroutines for expanding functionality. The Send context is invoked from the other contexts to send a message in response to a sensor reading or to route an incoming message.

The MATÉ architecture provides separation of contexts. One context cannot access the state of another context. There is only one single shared variable among the three contexts that can be accessed by special instructions. The context separation qualifies MATÉ to fulfill the traditional role of an OS. Compared to native TinyOS applications, the source code of MATÉ applications is much shorter.

12.3.3 TinyDB

TinyDB is a query processing system for extracting information from a network of TinyOS sensor nodes [11]. TinyDB provides a simple, SQL-like interface to specify the kind of data to be extracted from the network along with additional parameters, for example, the data refresh rate. The primary goal of TinyDB is to prevent the user from writing embedded C programs for sensor nodes or composing capsules of instructions regarding to MATÉ. The TinyDB framework allows data-driven applications to be developed and deployed much more quickly as developing, compiling, and deploying a TinyOS application.

Given a query specifying the data interests, TinyDB collects the data from sensor nodes in the environment, filters and aggregates the data, and routes it to the user autonomously. The network topology in TinyDB is a routing tree. Query messages flood down the tree and data messages flow back up the tree participating in more complex data query processing algorithms.

The TinyDB system is divided into two subsystems: sensor node software and a Java-based client interface on a PC. The sensor node software is the heart of TinyDB running on each sensor node. It consists of

- Sensor catalog and schema manager responsible for tracking the set of attributes, or types of readings and properties available on each sensor
- Query processor, utilizing the catalog to fetch the values of local attributes, to receive sensor readings from neighboring nodes, to combine and aggregate the values together, to filter, and to output the values to parents
- Small, handle-based dynamic memory manager
- Network topology manager to deal with the connectivity of nodes and to effectively route data and query subresults through the network

The sensor node part of TinyDB is installed on top of TinyOS on each sensor node as an application. The Java-based client interface is used to access the network of TinyDB nodes from a PC physically connected to a bridging sensor node. It provides a simple graphical query-builder and a result display. The Java API simplifies writing PC applications that query and extract data from the network.

12.3.4 SensorWare

SensorWare is a software framework for wireless sensor networks that provides querying, dissemination, and fusion of sensor data as well as coordination of actuators [12]. A SensorWare platform has less stringent resource restrictions. The initial implementation runs on iPAQ handhelds

(1 MB ROM/128 kB RAM). The authors intended to develop a software framework regardless of present sensor node limitations.

SensorWare developed at the University of California, Los Angeles, aims at the programmability of an existing sensor network after its deployment. The functionality of sensor nodes can be dynamically modified through autonomous mobile agent scripts. SensorWare scripts can be injected into the network nodes as queries and tasks. After injection, scripts can replicate and migrate within the network. Motivation for the SensorWare development was the observation that the distribution of updates and the download of complete images to sensor nodes are impractical for the following reasons. First, in a sensor network, a special sensor node may not be addressable because of missing node identifiers. Second, the distribution of complete images through a sensor network is highly energy consuming. Besides that, other nodes are affected by a download when multihop connections are necessary.

Updating complete images does not correspond to the low-power requirements of sensor networks. As a consequence, it is more practicable to distribute only small scripts. In the following section, the basic architecture and concepts of SensorWare are described in detail.

12.3.4.1 Basic Architecture and Concepts

SensorWare consists of a scripting language and a runtime environment. The language contains various basic commands that control and execute specific tasks of sensor nodes. These tasks include, for example, communication with other nodes, collaboration of sensor data, sensor data filtering, or moving scripts to other nodes. The language comprises necessary constructs to generate appropriate control flows.

SensorWare utilizes Tcl as scripting language. However, SensorWare extends Tcl's core commands. These core extension commands are joined in several API groups, such as Networking-API, Sensor-API, and Mobility-API (see Figure 12.9).

SensorWare is event-based. Events are connected to special event handlers. If an event is signaled an event handler serves the event according to its inherent state. Furthermore, an event handler is able to generate new events and to alter its current state by itself.

The runtime environment shown in Figure 12.10 contains fixed and platform-specific tasks. Fixed tasks are part of each SensorWare application. It is possible to add platform-specific tasks depending on specific application needs. The script manager task receives new scripts and forwards requests to the admission control task. The admission control task is responsible for script admission decisions and checks the overall energy consumption. Resource handlers manage different resources of the network.

Figure 12.11 shows the architecture of sensor nodes with included SensorWare software. The SensorWare layer uses OS functions to provide the runtime environment and control scripts. Static node applications coexist with mobile scripts. To realize dynamic programmability of a deployed sensor network a transient user can inject scripts into the network. After injection, scripts are replicated

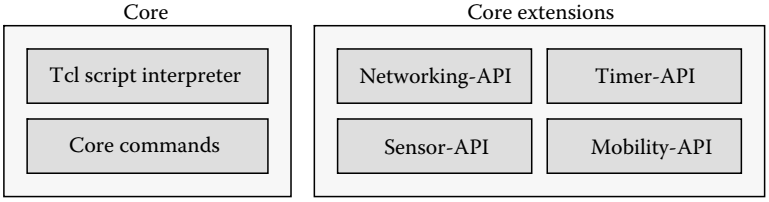


FIGURE 12.9 SensorWare scripting language.

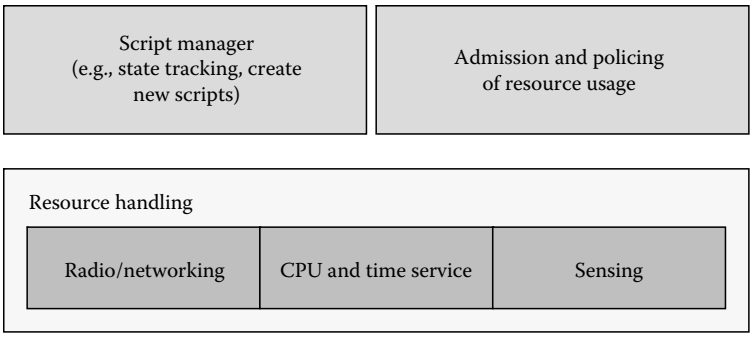


FIGURE 12.10 SensorWare runtime environment.

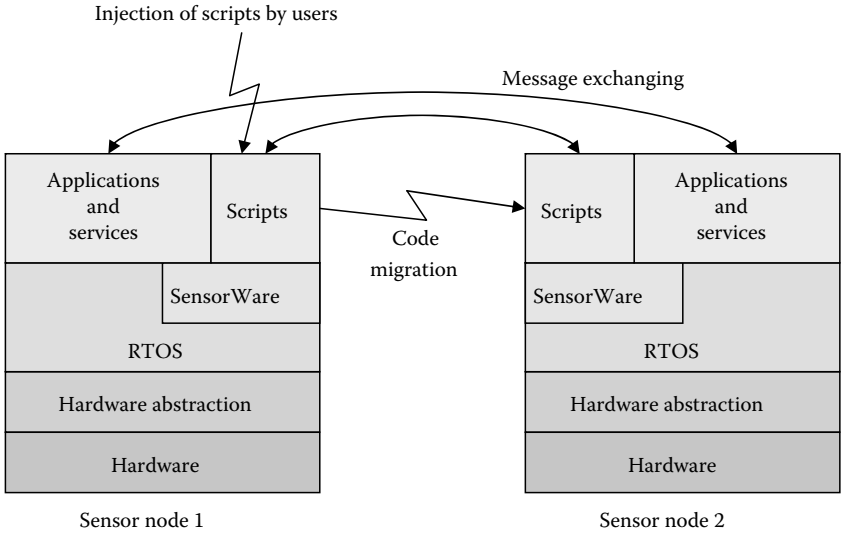


FIGURE 12.11 Sensor node architecture.

within the network and the script code migrates between different nodes. SensorWare ensures that no script is loaded twice onto a node during the migration process.

12.3.5 MiLAN

MiLAN is a middleware concept introduced by Mark Perillo and Wendi B. Heinzelman from the University of Rochester [13,14]. The main idea is to exploit the redundancy of information provided by sensor nodes. The performance of a cooperative algorithm in a distributed sensor network application depends on the number of involved nodes. Because of the inherent redundancy of a sensor network where several sensor nodes provide similar or even equal information, evaluating all possible sensor nodes leads to high energy and network costs. Therefore, a sensor network application has to choose an appropriate set of sensor nodes to fulfill application demands.

Each application should have the ability to adopt its behavior in respect to the available set of components and bandwidth within the network. This can be achieved by a parameterized sensor

node selection process with different cost values. These cost values are described by the following cost equations:

- *Application performance*: The minimum requirements for network performance are calculated from the needed reliability of monitored data. $F_R = \{S_i: \forall j \in J \quad R(S_i, j) \geq r_j\}$ where F_R stands for the allowable set of possible sensor node combinations, S_i represents the available sensor nodes, and $R(S_i)$ is their reliability.
- *Network costs*: Defines a subset of sensor nodes that meet the network constraints. The network feasible set $F_N = \{S_i: N(S_i) \leq n_0\}$ where $N(S_i)$ represents the total cost and n_0 the maximal data rate the network can support.
- Application performance and network costs are combined to the overall feasible set: $F = F_R \cap F_N$.
- Energy: Describes the energy dissipation of the network: $C_P(S_i) = \sum_{s_j \in S_i} C_P(S_j)$ where $C_P(S_j)$ is the power cost to node S_j.

It is up to the application to decide how these equations are weighted. This decision-making process is completely hidden from the application. Thus, the development process is simplified significantly. MiLAN uses two strategies to achieve the objective to balance QoS and energy costs:

- Turning off nodes with redundant information
- Using of energy-efficient routing

The MiLAN middleware is located between network and application layer. It can interface a great variety of underlying network protocols, such as Bluetooth and 802.11. MiLAN uses an API to abstract from network layer but gives the application access to low-level network components. A set of commands identifies and configures the network layer.

12.3.6 EnviroTrack

EnviroTrack is a TinyOS-based application developed at the University of Virginia that solves a fundamental distributed computing problem, environmental tracking of mobile entities [15]. Therefore, EnviroTrack provides a convenient way to program sensor network applications that track activities in their physical environment. The programming model of EnviroTrack integrates objects living in physical time and space into the computational environment of the application through virtual objects, called tracking objects. A tracking object is represented by a group of sensor nodes in its vicinity and is addressed by context labels. If an object moves in the physical environment, then the corresponding virtual object moves too because it is not bound to a dedicated sensor node. Regarding the tracking of objects, EnviroTrack does not assume cooperation from the tracked entity.

Before a physical object or phenomenon can be tracked, the programmer has to specify its activities and corresponding actions. This specification enables the system to discover and tag those activities and to instantiate tracking objects. For example, to track an object warmer than 100 °C, the programmer specifies a Boolean function, temperature >100 °C, a critical number or mass of sensor nodes, which fulfils the Boolean function within a certain time (a fact that is often referred to as freshness of information). These parameters of a tracking object are called aggregate state. All sensor nodes matching this aggregate state join a group. The network abstraction layer assigns a context label to this group. Using this label, different groups can be addressed independent of the set of nodes currently assigned to it. If the tracked object moves, nodes join or leave the group because of the changed aggregate state but the label resides persistent. This group management enables context-specific computation.

The EnviroTrack programming system consists of

- *EnviroTrack compiler*: In EnviroTrack programs, a list of context declarations is defined. Each definition includes an activation statement, an aggregate state definition, and a list of objects attached to the definitions. The EnviroTrack compiler includes C program templates. The whole project is then built using the TinyOS development tools.
- *Group management protocol*: All sensors associated to a group are maintained by this protocol. A group leader is selected out of the group members when the critical mass of nodes and freshness of the approximate aggregate state is reached. The group management protocol ensures that only a single group leader per group exists. The leader sends a periodical heartbeat to inform its members that the leader is alive. Additionally, the heartbeat signal is used to synchronize the nodes and to inform nodes that are not part of the group, but fulfils the sensing condition.
- *Object naming and directory services*: These services maintain all active objects and their locations. The directory service provides a way to retrieve all objects of a given context type. It also assigns names to groups so they can be accessed easily. It handles dynamical joining and leaving of group members.
- *Communication and transport services*: The migration transport protocol (MTP) is responsible for the transportation of data packets between nodes. All messages are routed via group leader nodes. Group leader nodes identify the context group of the target node and the position of its leader using the directory service. The packet is then forwarded to the leader of the destination group. All leadership information provided by MTP packets is stored in the leaders on a least recently used basis to keep the leader up-to-date and to reduce directory lookups.

EnviroTrack enables the construction of an information infrastructure for the tracking of environmental conditions. It manages dynamic groups of redundant sensor nodes and attaches computation to external events in the environment. Furthermore, EnviroTrack implements noninterrupted communication between dynamically changing physical locales defined by environmental events.

12.3.7 SeNeTs

SeNeTs is a middleware architecture for wireless sensor networks. It is developed at the University of Rostock [17]. The SeNeTs middleware is primarily designed to support the developer of a wireless sensor network during the predeployment phase (programming aspect). SeNeTs supports the creation of small and energy-saving programs for heterogeneous networks. One of the key features of SeNeTs is the optimization of APIs. The required configuration, optimization, and compilation of software components are processed by a development environment. Besides the programming aspect, the middleware supports the behavioral aspect as well, such as task change or evolution over time (Figure 12.4).

12.3.7.1 SeNeTs Architecture

SeNeTs is based on the software layer model introduced in Section 12.2. To increase flexibility and enhance scalability of sensor node software, it separates small functional blocks as shown in Figure 12.12. In addition, the OS layer is divided into a node-specific OS and a driver layer, which contains at least one sensor driver and several hardware drivers, such as timer driver and RF driver. The node-specific OS handles device-specific tasks, for example, boot-up, initialization of hardware, memory management, and process management as well as scheduling. Host middleware is the superior software layer. Its main task is to organize the cooperation of distributed nodes in the network. Middleware core handles four optional components, which can be implemented and exchanged

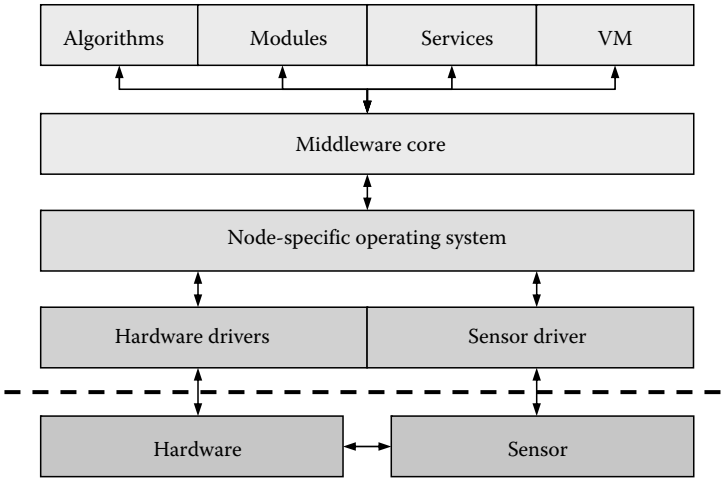


FIGURE 12.12 Structure of a node application.

according to the node's task. Modules are additional components that increase the functionality of the middleware. Typical modules are routing modules or security modules. Algorithms describe the behavior of modules. For example, the behavior of a security module can vary in the case the encryption algorithm changes. The services component contains the required software to perform local and cooperative services. This component usually cooperates with service components of other nodes to fulfill its task. VMs enable an execution of platform independent programs installed at runtime.

Figure 12.13 shows the expansion of the proposed architecture to a whole sensor network from the logical point of view. Nodes can only be contacted through services of the middleware layers. The distributed middleware coordinates the cooperation of services within the network. It is logically located in the network layer but physically exists in the nodes. All layers together in conjunction with their configuration compose the sensor network application. Thus, nodes do not perform any

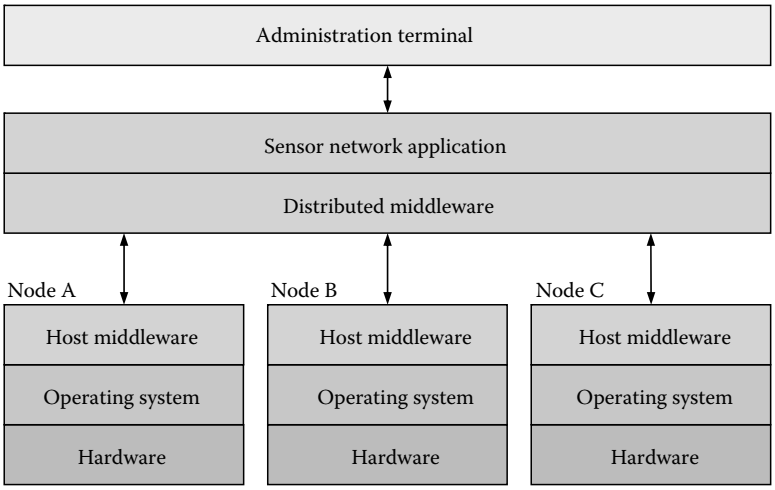


FIGURE 12.13 Structure of a sensor network.

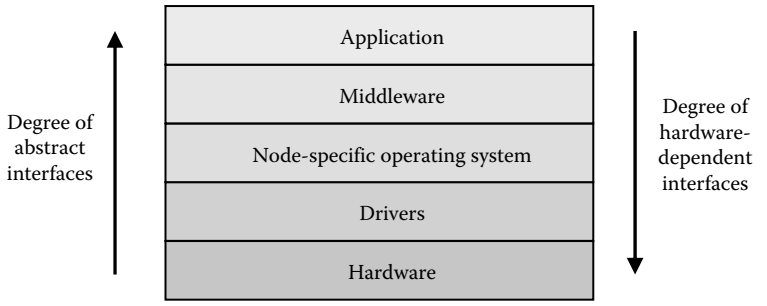


FIGURE 12.14 Interfaces within the software-layer model.

individual tasks. The administration terminal is an external entity to configure the network and evaluate results. It can be connected to the network at any location.

All functional blocks of the described architecture are represented by components containing real source code and a description about dependencies, interfaces, and parameters in XML. One functional block can be rendered by alternative components. All components are predefined in libraries.

12.3.7.2 Interface Optimization

One of the key features in SeNeTs is interface optimization. Interfaces are the descriptions of functions between two software parts. As illustrated in Figure 12.14, higher-level applications using services and middleware technologies require abstract software interfaces. The degree of hardware-dependent interfaces increases in lower software layers. Hardware-dependent interfaces are characterized by parameters to configure hardware components directly in contrast to abstract software interfaces whose parameters describe abstractions of the underlying system.

Software components require a static software interface to the application in order to minimize customization effort for other applications and to support compatibility. The use of identical components in different applications leads to a higher number of complex interfaces in these components. This is caused by component programming focused on supporting most possible use cases of all possible applications whereby each application uses only a subpart of the functionality of a component. Reducing the remaining overhead is the objective of generic software and can be done by interface optimization during compile time.

Interface optimizations result in proprietary interfaces within a node (Figure 12.15). Parts of the software cannot be exchanged without sensible effort. In a sensor node, the software is mostly

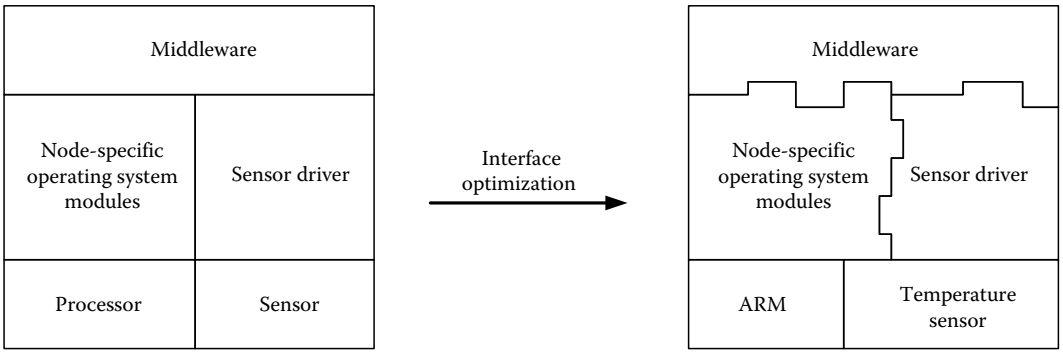


FIGURE 12.15 Interface optimization.

TABLE 12.1 Types of Interface Optimization

Optimization	Description
Parameter elimination	Parameters which are not used in one of the so-called subfunctions can be removed.
Static parameters	If a function is still called with same parameters, these parameters can be defined as constants or static variables in the global namespace. Thus, the parameter delivery to the function can be removed.
Parameter ordering	The sequence order of parameters is optimized in order to pass parameters through cascading ordering functions with same or similar parameters. It is particularly favorable in systems using processor registers instead of the system stack to deliver parameters to subfunctions.
Parameter aggregation	In embedded systems, many data types are not byte-aligned, for example, bits to configure hardware settings. If a function has several nonbyte-aligned parameters, these parameters may be combined.

static except programs for VMs. Accordingly, static linking is preferred. Statically linked software in conjunction with interface optimization leads to faster and smaller programs.

In SeNeTs, interfaces are customized to the application in contrast to common approaches used in desktop computer systems. These desktop systems are characterized by writing huge adaptation layers. The interface optimization can be propagated through all software layers and, therefore, saves resources. As an example of an optimization, a function OpenSocket(int name, int mode) identifies the network interface with its first parameter and the opening mode in the second parameter. However, a node that has only one interface opened with constant mode once or twice does not need these parameters. Consequently, knowledge of this information at compile time can be used for optimizing, for example, by

- Inlining the function or
- Eliminating both parameters from the delivery process

Another possibility is to change the semantics of data types. A potential use case is the definition of accuracy of addresses that results in changing data type's width. In SeNeTs, there are several types of interface optimizations proposed, which are given in Table 12.1.

Some optimizations such as "static parameters" are sometimes counterproductive in particular, if register-oriented parameter delivery is used. This is caused by the use of offset addresses once at parameter delivery instead of absolute addresses embedded in the "optimized" function. Consequently, the introduced optimizations strongly depend on

- Processor and processor architecture
- Type of parameter delivery (stack or register-oriented)
- Memory management (small, huge, and size of pointers)
- Objective of optimization (memory consumption, energy consumption, or compact code, etc.)
- Sensor network application

12.3.7.3 Development Process

Figure 12.16 shows the development process of sensor node software in SeNeTs. First, for each functional block the components have to be identified and included into the project. During design phase, the chosen components are interconnected and configured depending on developer's settings. Then, interface as well as parameter optimization is performed. The final source codes are generated and logging components can be included to monitor runtime behavior. The generated source codes are compiled and the executable is linked together. During evaluation phase, the created node application can be downloaded to the node and executed. Considering the monitoring results, a new design cycle

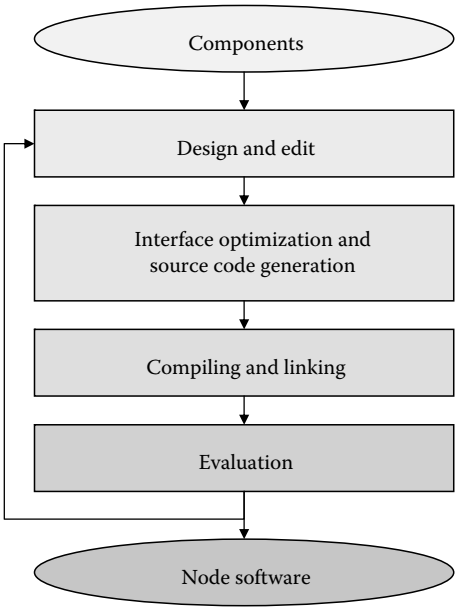


FIGURE 12.16 Development process of node software.

can be started to improve project settings. As a result of the design flow, optimized node application software is generated. The node application now consists of special tailored parts only needed by the specific application of the node.

Optionally, software components in a node can be linked together statically or dynamically. Statically linking facilitates an optimization of interfaces between several components within a node. A dynamic link process is used for components exchanged during runtime, for example, algorithms downloaded from other nodes. This procedure results in system-wide interfaces with significant overhead and prevents interface optimization.

12.3.8 Contiki

Contiki is a lightweight and flexible OS for 8 bit computers and integrated microcontrollers. It is built around an event-driven kernel [18]. Optional preemptive multithreading is provided as a linkable process library. Contiki was developed by Adam Dunkels et al. at the Swedish Institute of Computer Science (SICS). Dunkels also developed Protothreads which are extreme lightweight and stackless threads for memory-constrained systems and he developed a TCP/IP stack for embedded systems called uIP. Both, Protothreads and uIP, are part of Contiki.

The OS as well as its components are developed under an open-source license, which simplifies porting onto a wide range of computers and microcontrollers. This includes Texas Instruments MSP430 and Atmels AVR controllers which are used in a wide range of sensor networks [19].

12.3.8.1 Architecture

Contiki is divided into a core, containing fixed code, and a program part, containing reloadable components. These components are executed processes. Due to a few similarities to existing sensor network platforms, Contiki's core provides only a small functionality. Even the communication stack is implemented as a service on top of the kernel, although it belongs logically to the core. Therefore, Contiki uses an event-driven kernel.

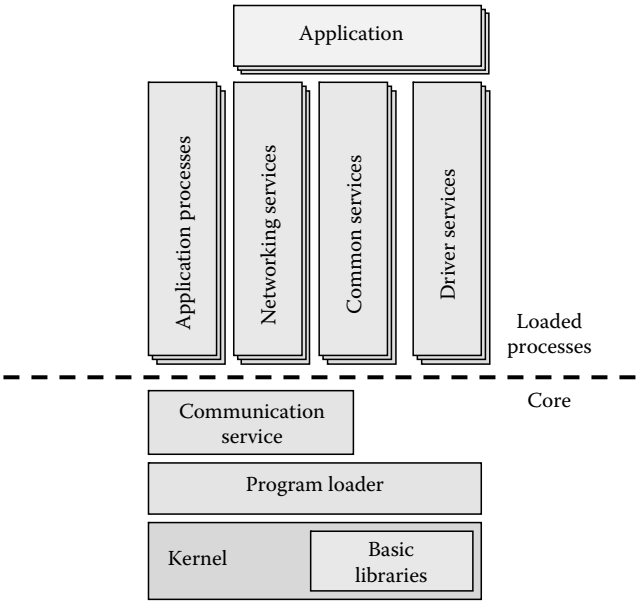


FIGURE 12.17 System architecture of Contiki.

Common systems using a multithreads require a memory stack for each thread that finally increases the total amount of memory. In addition, the required stack memory is often very large because it is hard to predict how much memory space is needed during lifetime. In contrast, the proposed event-driven model only needs to provide one stack while passing all enqueued events. Thus, all processes use the same stack. Furthermore, there is no need for mutual exclusion and locking of shared resources. Each event will run to completion before another event starts.

As it is illustrated in Figure 12.17, a working Contiki system consists of the kernel, the program loader, some basic libraries and the communication stack which is built as service on top of the kernel. Device drivers have to be implemented as services which can be used by applications or other services. All processes, i.e., services and applications, can communicate with each other through the kernel via message passing with the help of events. Also direct communication with the underlying hardware is provided by the kernel.

A process in Contiki is either an application or a service that provides functionality to other processes. Each process contains an event handler function to be called by the system if an event occurs. In addition, an optional polling handler function can be defined to poll events.

Contiki provides asynchronous as well as synchronous events. Synchronous events cause the target process to be scheduled immediately. Asynchronous events on the other hand are enqueued and dispatched to the target process after releasing the current thread. In-between the thread switch, asynchronous events are processed.

If a hardware interrupt occurs, it does not cause any event to be posted to prevent race conditions between the kernel and all threads. Instead, a polling flag is set by the kernel. This flag can be handled by all implemented polling handlers. Figure 12.18 illustrates the described interprocess operations, including service infrastructure.

12.3.8.2 Libraries

Contiki's kernel provides only the most essential features. To expand the functionality of Contiki, three kinds of methods are available. First, libraries are statically linked to the core (Figure 12.17).

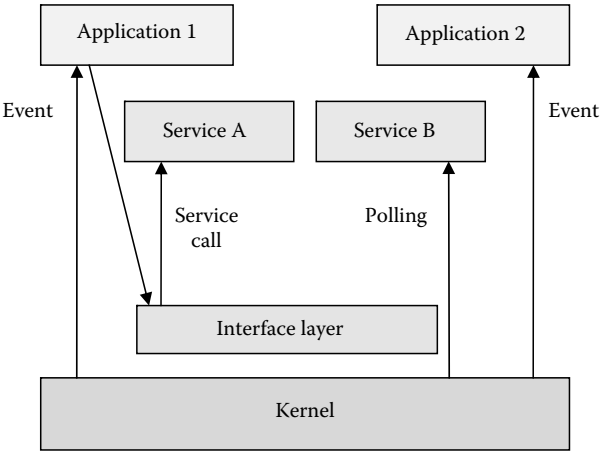


FIGURE 12.18 Interprocess operations in Contiki.

This method is recommended to expand the kernel only, such as to support memory management. Next, to add additional functionality to an application only, statical linking of specific libraries is appropriate. Third, linking of dynamical libraries is the preferred method to add functionality to the application depending on the current state. Dynamical libraries are loaded as services and therefore are processes, too. Hence, these processes can be accessed by all other applications as well. As illustrated in Figure 12.18, Contiki uses a service layer to provide applications access to the available services.

12.3.8.3 Runtime Programming

During lifetime of a sensor network, reprogramming of node's software may be required at runtime without recollecting nodes. TinyOS, for instance, allows reprogramming by using the bytecode interpreter MATÉ (see Section 12.3.2) which can be installed as one program on top of the OS. Contiki does not need such an interpreter, instead, it provides loading and unloading of all kinds of processes even drivers and, for instance, the communication stack, at any time. Therefore to support OTA programming, a specialized protocol is defined.

12.3.8.4 Platforms

Contiki was developed as lightweight OS for sensor networks. Therefore, Contiki has a very small memory footprint and uses all constraint processing resources as efficient as possible. Currently, the system is ported to a lot of target systems, for instance, MSP430 and Atmel AVR.

Due to Contiki's modularity, porting the system affects only some parts. These parts are the boot-up code, architecture-specific parts of the program loader, and device drivers. If multithreading should be available on the target architecture, the stack switching code of the multithreading library has to be adopted, too. But there is no need for changing the kernel and the service layer. Thus, porting Contiki to a new target takes only a few hours or days.

12.3.9 Sensor Web Enablement

The standards organization Open Geospatial Consortium Inc.® (OGC) [20] has published several standards and specifications for geospatial and location-based services on the Web. This consortium

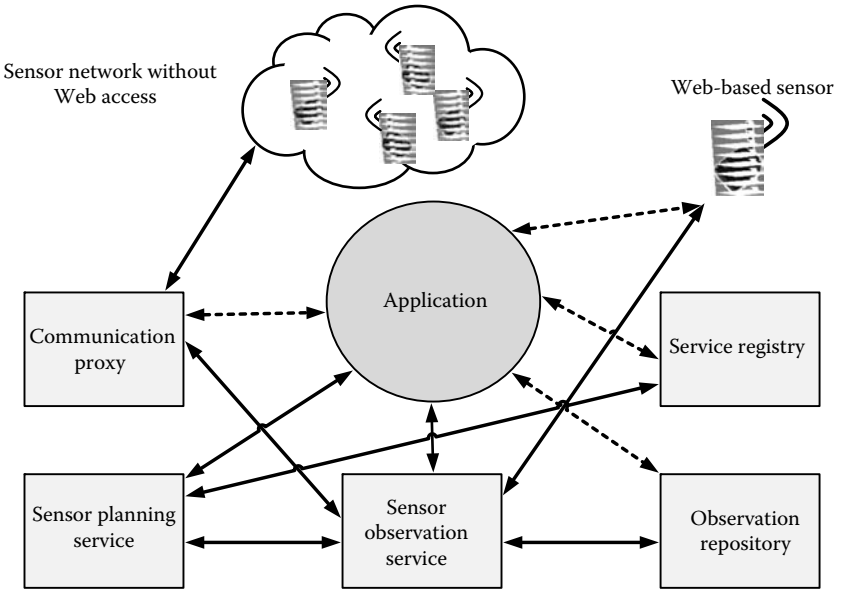


FIGURE 12.19 Simplified communication structure for networks based on SWE.

has more than 350 member companies. Some of these important partners are Google, Microsoft, NASA, and ORACLE.

The Sensor Web Enablement (SWE) is one project, with the objective to discover and access sensor nodes by standardized interfaces. The defined specifications describe sensor data types, discovery procedures, and communication stacks. Figure 12.19 shows the simplified communication structure of a SWE network. The dotted lines show the direct and near real-time communication channels. The integration of very small or non-Web-based devices with a communication proxy is also feasible. The common client interface allows a single interface for applications to access all sensor nodes which are compatible with SWE.

Currently, SWE is divided into seven candidate specifications. The five main specifications are presented in Sections 12.3.9.1 through 12.3.9.5. The description is based on XML which allows processing from any kind of sensor data based on time and location of any phenomenon.

12.3.9.1 Observations and Measurements

The observations and measurements (O&M) specification is a flexible aggregation of standard constructs that provide description, exchanging, and planning of measurements and observations. Due to the extensibility of XML, unknown devices or physical effects as well as their units can be flexibly described. The O&M specification defines an observation as an event describing a phenomenon. It contains triggering time, current position, and the device detecting the event.

12.3.9.2 Sensor Model Language

The sensor model language (SensorML) is a model for discovery, exploration, and exploitation of sensor nodes and sensor observations. With SensorML it is possible to describe all kinds of measurements and post-measurement processings as a process model in XML. Therefore, a description of a sensor node is a functional model of different process models. Each model has standard inputs, outputs, and parameters which can be used by applications for automated exploration of process models. Furthermore, additional metadata is included to enable discovery and human assistance.

12.3.9.3 Transducer Markup Language

The transducer markup language (TML) describes protocols to communicate with sensor systems using a XML model. It supports real-time streaming and command exchange from and to sensor nodes. Hence, it defines response characteristics of a transducer and methods for transportation and preparation of sensor data also in spatial and temporal associations. Through TML definitions, different phenomena can be correlated to a function to reduce jitter, artifacts, or noise in the data.

12.3.9.4 Sensor Observation Service

The sensor observation service (SOS) defines a standard and consistent interface to access data. The SOS communicates via a handshake protocol between a client and an observation repository. The observation repository is a database anywhere in the network to collect measurement data which can be accessed by any clients.

Only for soft real-time sensor channels, a direct connection between applications and sensors is available. To discover sensors, each SOS has an entry in a global service registry anywhere in the network with characteristic description of the sensor node.

12.3.9.5 Sensor Planning Service

The sensor planning service (SPS) is an interoperable web service interface that allows planning and execution of data collections from one or more sensors. Standard interfaces are defined to request information about the capabilities of a SPS. These interfaces consist of request definitions, status inquiries, updating or canceling of requests, and further OGC Web services that provide access to more complex processing algorithms.

12.4 Simulation, Emulation, and Test of Large-Scale Sensor Networks

Applications and protocols for wireless sensor networks require novel programming techniques and new approaches for validation and test of sensor network software. In practice, sensor nodes have to operate in an unattended manner. The key factor of this operation is to separate unnecessary information from important ones as early as possible in order to avoid communication overhead. In contrast, during implementation and test phases, developers need to obtain as much information as possible from the network. A test and validation environment for sensor network applications has to ensure this.

Consider a sensor network with thousands of sensor nodes. Furthermore, consider developing a data fusion and aggregation algorithm that collects sensor information from nodes and transmits them to few base stations. During validation and test, developers often have to change application code, recompile, and upload a new image onto the nodes. These updates often result in flooding of the network using the wireless channel. However, this would dissipate a lot of time and energy. But how could we ensure that every node runs the most recent version of our application?

Pure simulation produces important insights. However, modeling the wireless channel is difficult. Simulation tools often employ simplified propagation models in order to reduce computational efforts for large-scale networks. Widely used simulation tools, such as ns-2 [21] use simplified network protocol stacks and do not simulate at a bit level. Furthermore, code used in simulations often cannot be reused on real sensor node hardware; why should developers implement applications and protocols twice?

In contrast to simulation, implementation on a target platform is often complicated. The targeted hardware itself may be still in development stage. Perhaps there are a few prototypes, but developers

need hundreds of them for realistic test conditions. Moreover, prototype hardware is very expensive and far away from the targeted "1 cent/node." Consequently, a software environment is required that combines the scaling power of simulations with real application behavior. Moreover, the administration of the network must not affect sensor network applications. Afterwards, three current software approaches are presented.

12.4.1 A TinyOS SIMulator

Fault analysis of distributed sensor networks or their particular components is quite expensive and time consuming especially when sensor networks consist of hundreds of nodes. For that purpose, a simulator providing examination of several layers (e.g., communication layer, routing layer) is an efficient tool for sensor application development.

TinyOS SIMulator (TOSSIM) is a simulator for wireless sensor networks based on the TinyOS framework. As described in Refs. [23,24], the objectives of TOSSIM are scalability, completeness, fidelity, and bridging. Scalability means TOSSIM's ability to handle large sensor networks with many nodes in a wide range of configurations. The reactive nature of sensor networks requires not only the simulation of algorithms but also the simulation of complete sensor network applications. Therefore, TOSSIM achieves completeness in covering as many system interactions as possible. TOSSIM is able to simulate thousands of nodes running entire applications. The simulator's fidelity becomes important for capturing subtle timing interactions on a sensor node and between nodes. A significant attribute is the revealing of unanticipated events or interactions. Therefore, TOSSIM simulates the TinyOS network stack down to bit level. At last, TOSSIM bridges the gap between an academic algorithm simulation and a real sensor network implementation. Therefore, TOSSIM provides testing and verification of application code that will run on real sensor node hardware. This avoids programming algorithms and applications twice, for simulation and for deployment. The TOSSIM components are integrated into the standard TinyOS compilation tool chain which supports the direct compilation of unchanged TinyOS applications into the TOSSIM framework.

Figure 12.20 shows a TinyOS application divided into hardware-independent and hardware-dependent components. Depending on the target platform, the appropriate hardware-dependent modules are selected in the compilation step. This permits an easy extension regarding new sensor node platforms. At the same time, this is the interface to the TOSSIM framework. Compared with a native sensor node platform, TOSSIM is a sensor node emulation platform supporting multiple sensor node instances running on standard PC hardware. Additionally, the TOSSIM framework includes a discrete event queue, a small number of reimplemented TinyOS hardware abstraction components, mechanisms for extensible radio and analog-to-digital converter (ADC) models, and communication services for external programs to interact with a simulation.

The core of the simulator is the event queue. Because TinyOS utilizes an event-based scheduling approach, the simulator is event-driven, too. TOSSIM translates hardware interrupts into discrete simulator events. The simulator event queue emits all events that drive the execution of a TinyOS application. In contrast to real hardware interrupts, events cannot be preempted by other events and therefore are not nested.

The hardware emulation of sensor node components is performed by replacing a small number of TinyOS hardware components. These include the ADC, the clock, the transmit strength variable potentiometer, the EEPROM, the boot sequence component, and several components of the radio stack. This enables simulations of a large number of sensor node configurations.

The communication services are the interface to PC applications driving, monitoring, and actuating simulations by communicating with TOSSIM over TCP/IP. The communication protocol was designed at an abstract level and enables developers to write their own systems that hook into TOSSIM. TinyViz is an example of a TOSSIM visualization tool that illustrates the possibilities

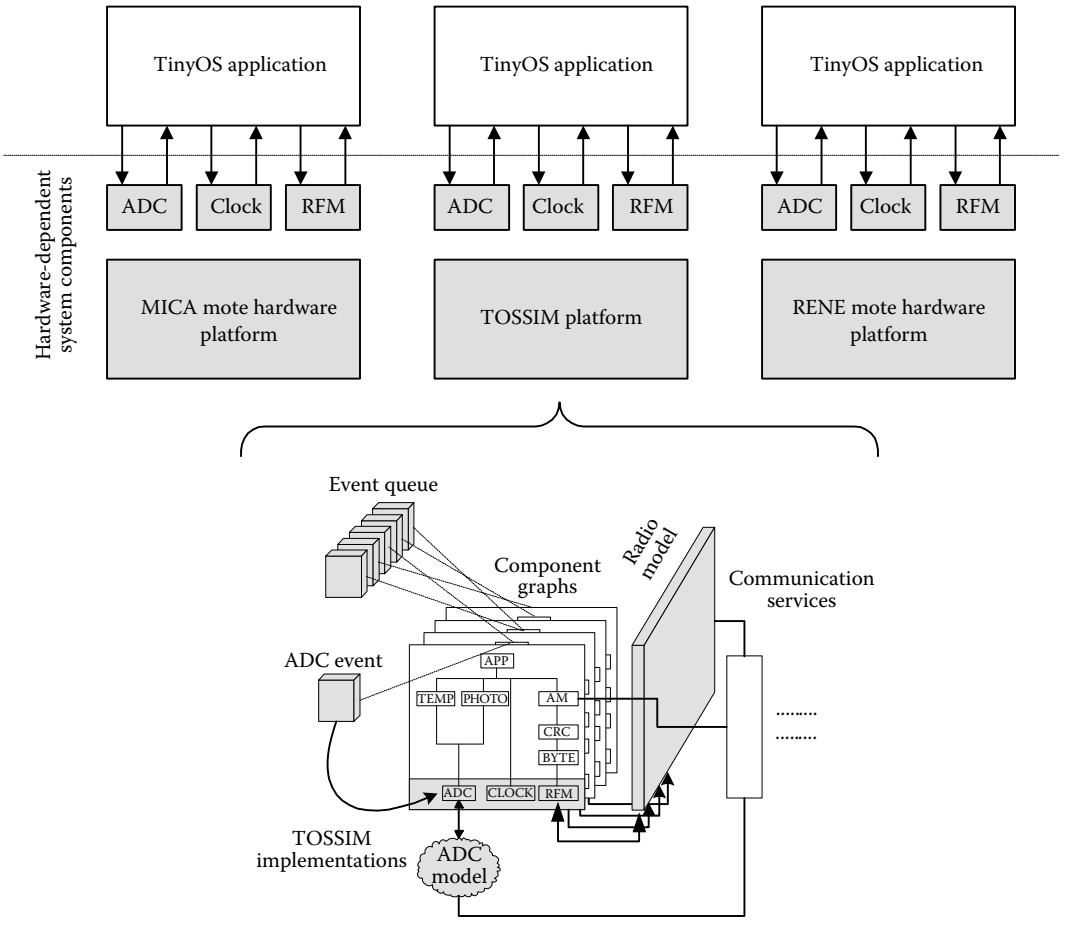


FIGURE 12.20 Comparison of TinyOS and TOSSIM system architecture.

of TOSSIM's communication services. It is a Java-based graphical user interface providing visual feedback on the simulation state to control running simulations, for example, modifying ADC readings and radio loss properties. A plug-in interface for TinyViz allows developers to implement their own application-specific visualization and control code.

TOSSIM does not model radio propagation, power draw, or energy consumption. TOSSIM's is limited by the interrupts that are timed by the event queue. They are nonpreemptive. In conclusion, TOSSIM is an event-based simulation framework for TinyOS-based sensor networks. The open-source framework and the communication services permit an easy adaptation or integration of simulation models and the connection to application-specific simulation tools.

12.4.2 EmStar

EmStar is a software environment for developing and deploying applications for sensor networks consisting of 32 bit embedded Microserver platforms [25,26]. EmStar consists of libraries, tools, and services. Libraries implement primitives of interprocess communication. Tools support simulation, emulation, and visualization of sensor network applications. Services provide network functionality, sensing, and synchronization.

TABLE 12.2 EmStar Components

Component	Description
EmRun	Management and watchdog process (responsible for start-up, monitoring, and shutdown of EmStar modules)
EmProxy	Gateway to a debugging and visualization system
MicroDiffusion, udpd, LinkStats, neighbors	Network protocol stack for wireless connections
Timehist, syncd, audiod	Audio sampling service
FFT, detect, collab_detect	"Hypothetical modules," responsible for Fast Fourier Transformation, and (collaborative) event detection

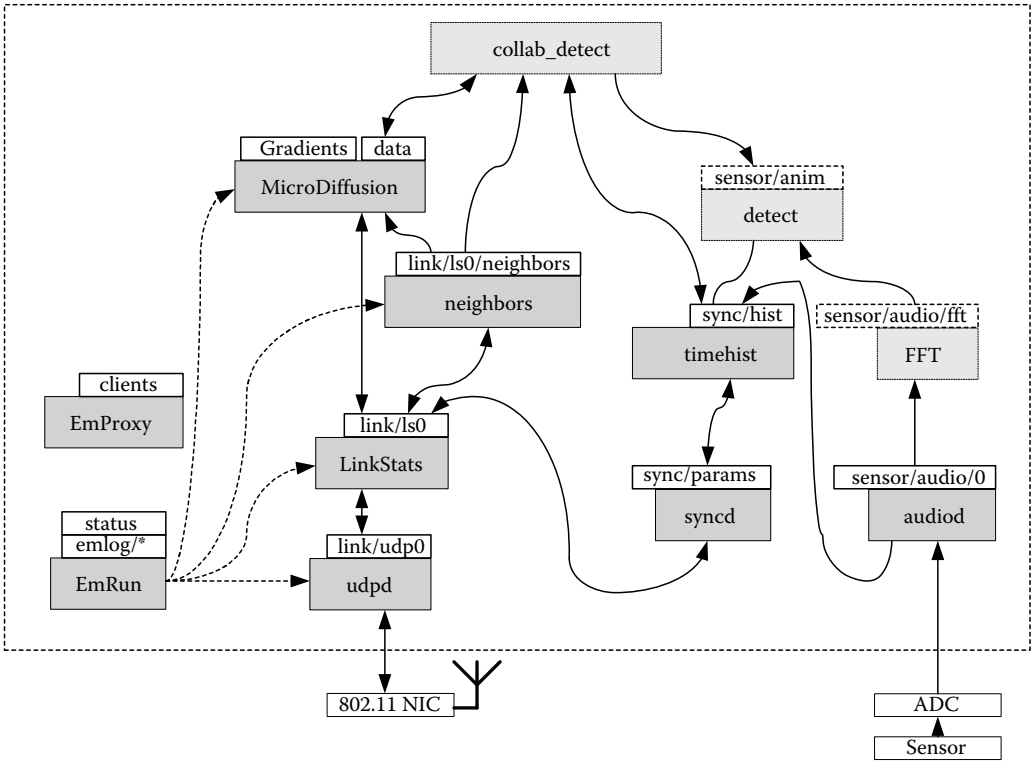


FIGURE 12.21 EmStar sample application consisting of EmStar core modules (dark-gray) and hypothetical application-specific modules (light-gray).

EmStar's target platforms are so-called Microservers, typically iPAQ or Crossbow Stargate devices. EmStar does not support Berkeley Motes as platform, however, can easily interoperate with Motes. EmStar consists of various components. Table 12.2 gives the name and a short description of each component. The last row of the table contains hypothetical, application-specific components; all others are EmStar core components.

Figure 12.21 illustrates the cooperation of EmStar components in a sample application for environmental monitoring. The dark-gray boxes represent EmStar core modules. Hypothetical application-specific modules are filled light gray. The sample application collects data from an audio sensor and therewith tries to detect the position of an animal in collaboration with neighboring sensor nodes.

12.4.2.1 EmStar Tools and Services

EmStar provides tools for the simulation, emulation, and visualization of a sensor network and its operation. EmSim runs virtual sensor nodes in a pure simulation environment. EmSim models both

radio and sensor channels. EmCee runs the EmSim core, however, uses real radios instead of modeled channels. Both EmSim and EmCee use the same EmStar source code and associated configuration files as a real deployed EmStar system. This aspect alleviates the development and debugging process of sensor network applications.

EmView is a visualization tool for EmStar systems that uses a UDP protocol to request status updates from sensor nodes. In order to obtain sensor node or network information, EmView queries an EmProxy server that runs as part of a simulation or on a real node. EmRun starts, stops, and manages an EmStar system. EmRun supports process respawn, in-memory logging, fast startup, and graceful shutdown.

EmStar services comprise link and neighborhood estimation, time synchronization, and routing. The neighborhood service monitors links and maintains lists of active, reliable nodes. EmStar applications can use these lists to get informed about topology changes. The LinkStats service provides applications with more detailed information about link reliability than the Neighborhood service, however, produces more packet overhead. Multipath-routing algorithms can benefit from the LinkStats service by weighting their path choices with LinkStats information. The TimeSync service is used to convert timestamps between different nodes. Additionally, EmStar supports several routing protocols, but allows the integration of new routing protocols as well.

12.4.2.2 EmStar IPC mechanism

Communication between EmStar modules is managed by so-called framework for user-space devices (FUSD)-driven devices, a microkernel extension to Linux. FUSD allows device file callbacks to be proxied into user space and implemented by user space programs instead of kernel code. Besides intermodule communication, FUSD allows interaction of EmStar modules and users. FUSD drivers are implemented in user space, however, can create device files with the same semantics as kernel-implemented device files. Applications can use FUSD-driven devices to transport data or expose state.

Several device patterns exist for EmStar systems that are frequently needed in sensor network applications. Example device patterns comprise a status device pattern exposing the current state of a module, a packet device pattern providing a queued multiclient packet interface, a command device pattern that modifies configuration files and triggers actions, and a query device pattern implementing a transactional RPC mechanism.

12.4.3 SeNeTs—Test and Validation Environment

In SeNeTs, SNAs run distributed on independent hosts such as PCs, PDAs, or evaluation boards of embedded devices [27]. The parallel execution decouples applications and simulation environment. The quasi-parallel and sequential processing of concurrently triggered events in simulations is disadvantageous compared with real-world programs. SeNeTs prevents this very unpleasant effect. Without SeNeTs, this effect results in sequenced execution of parallel working SNAs and corrupted simulation output. To summarize, realistic simulations of sensor networks are complicated.

12.4.3.1 System Architecture

The development and particularly the validation of distributed applications are hard to realize. Especially systems with additional logging and controlling facilities affect the primary behavior of applications. Suppose, a logging message is transmitted, then an application message may be transmitted delayed. Exceptionally in wireless applications with limited channel capacity, the increased communication leads to a modified timing behavior and as a consequence to different results.

The channel capacity is defined as $n^{-0,5}$, since n symbolizes the number of nodes. Due to the degrading channel capacity in large sensor networks, the transport medium acts as a bottleneck [28].

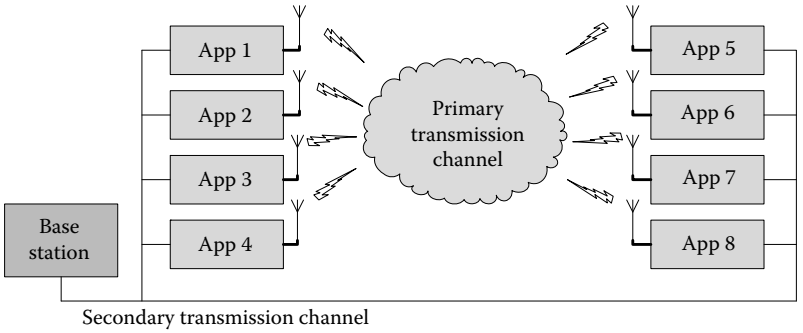


FIGURE 12.22 Communication channels in SeNeTs.

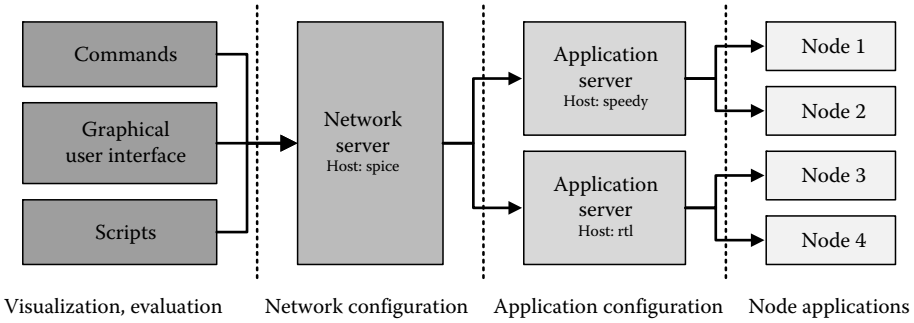


FIGURE 12.23 SeNeTs components using the secondary transmission channel.

Thus, in wireless sensor networks with thousands of nodes, the bottleneck effect becomes the dominant part.

To eliminate the bottleneck effect, SeNeTs contains two independent communications channels as illustrated in Figure 12.22. The primary communication channel is defined by the sensor network application. It uses the communication method required by SNAs, for example, Bluetooth or ZigBee. The secondary communication channel is an administration channel only used by SeNeTs components. This channel transmits controlling and logging messages. It is independent of the primary communication channel and uses a different communication method, for example, Ethernet or Ultrasound. The separation into two communication channels simplifies the decoupling of application modules and administration modules after testing.

The parallel execution of applications on different host systems requires a cascaded infrastructure to administrate the network. Figure 12.23 displays important modules in SeNeTs: node applications, application servers (ASs), a network server (NS), and optional evaluation or visualization modules. All of these modules are connected via the secondary transmission channel.

12.4.3.2 Network Server

The NS administrates sensor networks and associated sensor nodes. The NS starts, stops, or queries SNAs. In an SeNeTs network, exactly one NS exists. However, this NS is able to manage several sensor networks simultaneously. Usually, the NS runs as service of the OS.

A NS opens additional communication ports. External programs, such as scripts, websites, or telnet clients can connect to these ports to send commands. These commands may be addressed and forwarded to groups or stand-alone components. Furthermore, the NS receives logging messages from

applications containing their current state. Optional components, such as graphical user interfaces can install callbacks to receive this information.

12.4.3.3 Application Server

The AS manages instances of node applications on one host (Figure 12.23). It acts as bridge between node applications and the NS. Usually, at least one AS exists within the SeNeTs network. Ideally, only one node application should be installed on an AS to prevent quasiparallel effects during runtime.

The AS runs independent of the NS. It connects to the NS via a pipe to receive commands. Each command is multiplexed to one of the connected node applications. Moreover, if the pipe to the NS breaks, node applications will not be affected besides losing logging and controlling facilities. Later, the NS can establish the pipe again.

Generally, an AS starts as service together with the host's OS. At startup, it requires configuration parameters of the node's hardware. With these parameters, the AS assigns hardware to node applications. Suppose a host system that comprises two devices representing sensor nodes as principally shown in Figure 12.23. Then, the AS requires device number, physical position of the node, etc. to configure the dynamically installed node applications at runtime.

12.4.3.4 SeNeTs Application

Applications for wireless sensor nodes are usually designed based on a layered software model as depicted in Figure 12.24a [17]. On top of the node's hardware, a specialized OS is set up such as TinyOS [22]. A sensor driver contains software to initialize the measurement process and to obtain sensor data. Above the OS and the sensor driver, middleware components are located containing services to aggregate data or to determine the node's position. The aforementioned modular design allows:

- Abstraction of hardware, for example, sensors, communication devices, memory, etc.
- Adaptation of the node's OS
- Addition of optional components, e.g., logging and configuration

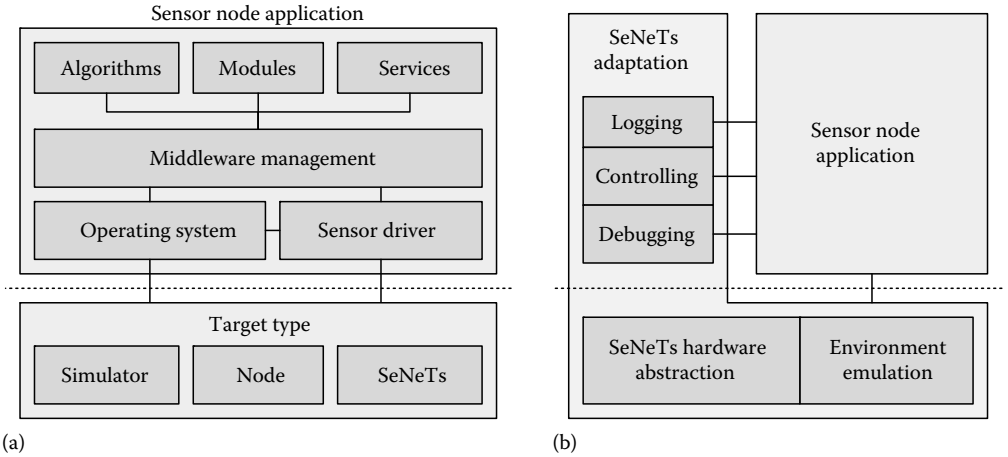


FIGURE 12.24 (a) Software layer model of a SNA and (b) Software layer model of a SeA.

The SeNeTs Adaptation is a set of components which are added or exchanged to wrap the SNA. Figure 12.24b represents the SeNeTs Adaptation layer consisting of at least a logging component, a controlling unit, a HAL, and an optional environment encapsulation module. These additional components provide substantial and realistic test and controlling facilities.

An application composed of a SNA and SeNeTs Adaptation components is called sSeNeTs Application (SeA). The SNA is not changed by added components. Generally, it is not necessary to adapt the SNA to SeNeTs interfaces. However, supplementary macros can be added to interact with the linked components.

A SeNeTs application runs as process of the host system. Due to the architecture of a SNA with its own OS, the SeNeTs application runs autonomously without interaction to other processes of the host system. At startup, the SeNeTs application opens a pipe to communicate with the AS. After the test phase, all SeNeTs components can be removed easily by recompilation of all node applications. SeNeTs specific components and logging calls are automatically deactivated due to compiler switches.

12.4.3.5 Environment Management

Sensor network applications require valid environment data, such as temperature or air pressure. Under laboratory conditions, this information is not or partly not available. Therefore, environment data must be emulated. SeNeTs provides these environment data to the node application by the environment emulation module (Figure 12.24b). All environment emulation modules are controlled by the environment management of the NS which contains all predefined or configured data (Figure 12.25). This data comprises positions of other nodes, distances to neighboring nodes, etc. If required, other data types may be added. In the AS, the environment data cache module stores all environment information required by each node application to reduce network traffic.

Optionally, position-based filtering is provided by the environment emulation component of SeNeTs. Especially, if large topologies of sensor nodes should be emulated under small-sized laboratory conditions, this filtering approach is essential. Suppose real and virtual positions of nodes are known, a mapping from physical address to virtual address is feasible. A node application only receives messages from nodes that are virtually in transmission range. All other messages are rejected by the SeNeTs Adaptation components. This is accomplished by setting up a filter in the primary communication channel.

One application scenario that illustrates position-based filtering is flood prevention. Here, sensor nodes are deployed in sandbags, piled along a dike of hundreds of meters or even kilometers. These nodes measure the humidity and detect potential leakages. Testing this scenario under real-world conditions is not practical and very expensive. However, the evaluation under real-world conditions

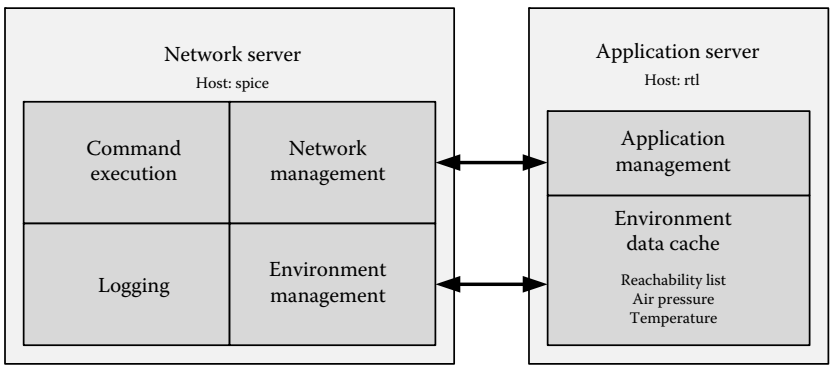


FIGURE 12.25 Environment management in SeNeTs.

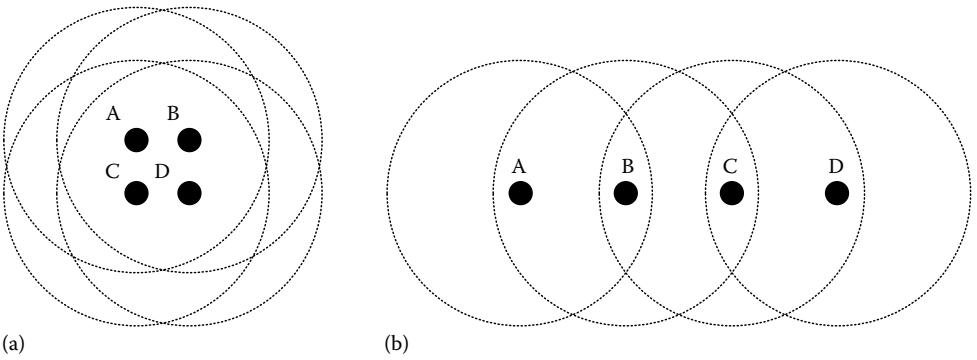


FIGURE 12.26 (a) Physically arranged sensor nodes (solid circles). All nodes are in transmission range (dotted line) of each other. (b) Virtually arranged nodes with appropriated transmission ranges. Nodes are not longer able to communicate without routing.

of software regarding communication effort, self-organization of the network, routing, and data aggregation is most important.

Figure 12.26 illustrates the difference between laboratory and real world. The left side represents laboratory conditions where all nodes are in transmission range of each other. The right side sketches the flood prevention scenario under real conditions. On the left side, the nodes A–D are in transmission range of each other. Therefore in contrast to the real-world scenario, no routing is required. Next, data aggregation yields wrong results, because nodes are not grouped as they would in reality. Thus, if physically arranged nodes in test environment do not meet the requirements of the real world, the results are questionable.

Assume, node A sends a message to node D, then all nodes receive the message due to the physical vicinity in the test environment (Figure 12.26a). Nodes C and D receive the message, but they are not in the virtual transmission range of node A. Thus, the environment emulation module rejects these messages. As a result, SeNeTs prevents a direct transmission from node A to node D. Messages can be transmitted only by using routing nodes B and C (Figure 12.26b). In short, the emulation of the sensor network software becomes more realistic.

12.4.4 J-Sim

J-Sim is primarily designed to simulate any network composed of nodes [29]. The simulator is based on autonomous component architecture (ACA). On top of the ACA, a packet switched internetworking framework (INET) is defined. The component design in J-Sim is very similar to an integrated chip (IC) design architecture where each chip is defined as stand-alone module communicating with other chips via wires. Due to their simple interface (port), components can be designed, developed, and tested individually. Each port contains a contract which describes the type of information exchanged with other components to ensure the statical-type safety.

The simulator is written in Java which is also used to compile components. Supplementary, Tcl is applied to control the simulation process via scripts. Therefore, it is platform-independent and extensible. At runtime, all required components are instantiated and their ports are connected via Tcl scripts. The runtime environment of the simulator and each instance of components are inserted in a virtual directory tree which can be accessed by any simulation element or other scripts.

To simulate sensor networks, J-Sim provides a specialized add-on called Sensorsim framework [30] using ACA and INET. It requires only a fractional amount of memory to simulate correctly compared

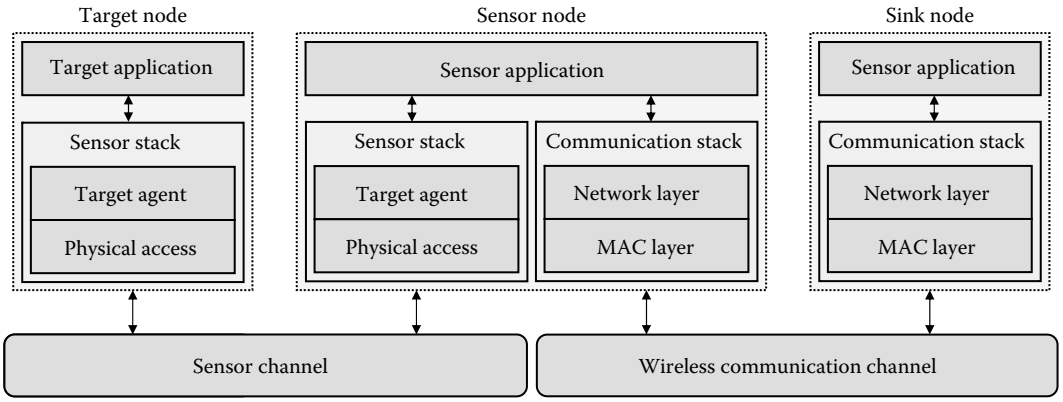


FIGURE 12.27 Model of a sensor network using SensorSim in J-Sim.

to other network simulators, such as ns-2, which often crashes throwing out of memory exceptions in large networks with thousands of nodes. The Sensorsim framework additionally defines

- Three types of nodes (sink node, target node, and sensor node)
- Sensor models
- Wireless communication channels
- Environmental models (mobility and power modi)

The sensor network in SensorSim is modeled as visualized in Figure 12.27. The wireless communication channel defines the communication between sink nodes and other sensor nodes. In contrast, the sensor channel is used to detect signals such as positions or measured data.

All nodes are additionally linked with power models to estimate the power consumption precisely. Further, the framework supports components to track nodes using a NodePositionTracker or to emulate routing protocols such as ad hoc on-demand distance vector routing (AODV) [31] or geometric routing such as greedy perimeter stateless routing (GPSR) [32].

The presented framework SensorSim of J-Sim provides a very flexible way to configure a realistic scenario of wireless sensor networks. As an alternative, GloMoSim may be used to simulate wireless networks [33]. In contrast to J-Sim and GloMoSim, the simulation environment OPNET uses a hierarchical arrangement of components as used in the Internet to simulate networks [34].

12.5 Mastering Deployed Sensor Networks

Development of sensor network applications is a challenging task. As a consequence of deploying large-scale networks, an administrator of a sensor network may need to monitor areas of interest, and usually wants to control the deployed network (Figure 12.28). While specific single sensor information has no high relevance, trends, histories, and variances are very important.

Especially nontechnicians should be able to monitor and administrate their applications easily. Powerful graphical user interfaces are required to simplify deployment, control, and monitor large-scale wireless sensor networks in their application domains such as logistics, safety systems, and environment applications. These software mastering environments are usable for application developers, administrators, and maintainers, too. Typical tasks of these tools are

- Deployment and initial setup of sensor nodes
- Easy configuration of the whole network or few sensor nodes as well

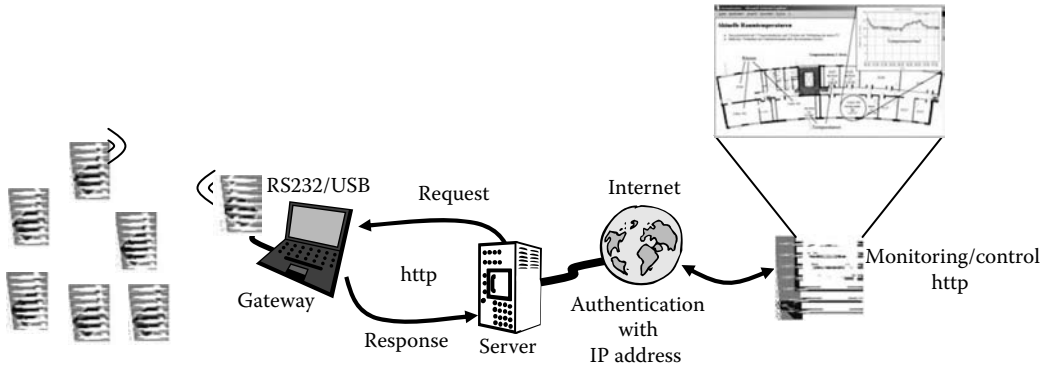


FIGURE 12.28 A deployed wireless sensor network.

- Full graphical overview of connected sensor nodes
- Monitoring and analyzing of measured parameters with charts, histograms, topologies
- Life cycle management to support future requirements
- OTA programming
- Management of various sensor networks

Software mastering deployed sensor networks is typically a distributed software system. It is mostly divided into three main parts—a so-called multitier architecture (Figure 12.29). This architecture is used, for example, by Moteworks [35], the Tiny Application Sensor Kit (TASK) [36], and EnviSense [37].

The mote network tier includes software components implementing routing and OTA flashing facilities on sensor nodes. Protocols used at the mote tier are XMesh (Moteworks) or the Collection Tree Protocol (TinyOS 2.0) to provide routing. To support OTA, Deluge or XOTAP (Moteworks) can be applied. The connection to other networks is mostly established by gateways, for example, the

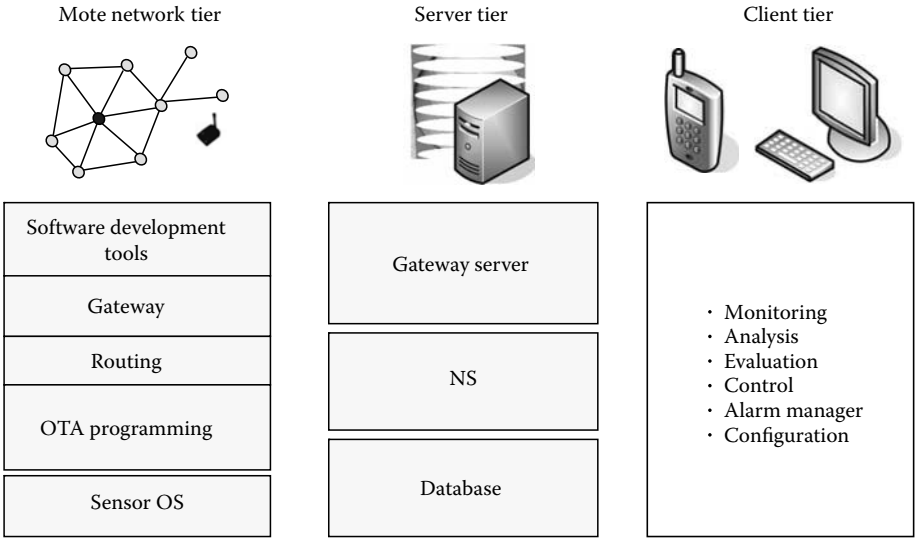
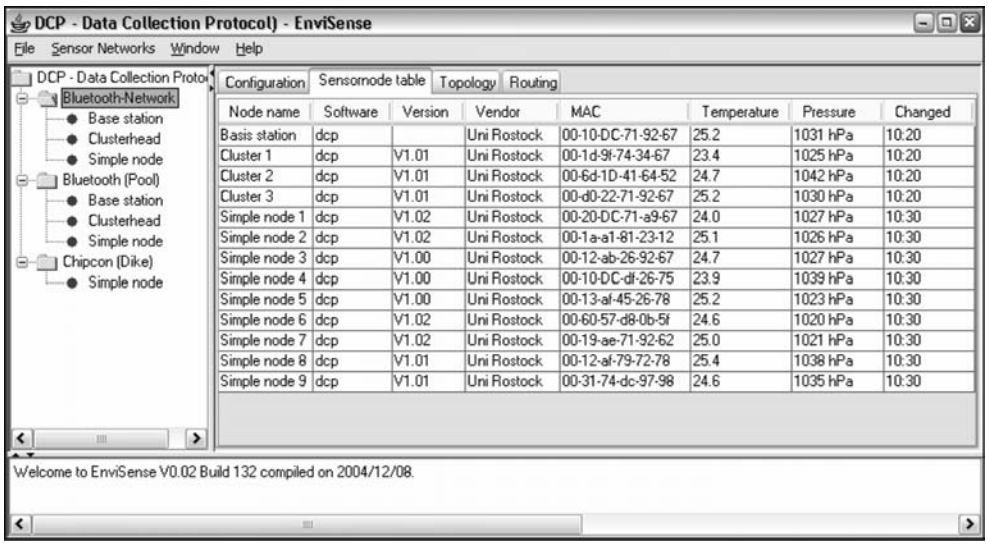


FIGURE 12.29 Multitier model of software mastering tools.

TASKServer used in the Tiny Application Sensor Kit. These gateways act as a proxy for the whole sensor network to the Internet.

The server tier links sensor networks to enterprise networks where gathered data is preprocessed and filtered by logging servers containing packet parsers as used in XServe (Moteworks). The data is typically stored in SQL databases, such as TASK DBMS (TASK), PostgreSQL (Moteworks), or mySQL. On this way, sensor readings, sensor network health statistics, sensor locations, and calibration coefficients can be approximated and evaluated by various clients.

The client tier has access to the databases and uses graphical tools to monitor, analyze, and visualize sensor data. It is the interface layer between the user and the deployed sensor network. Envisense, for example, uses a graphical front-end (Figure 12.30) similar to MoteView to visualize available information and to control the whole deployed network. TASK uses Task Field Tools on a PDA for in situ diagnosis of a deployed sensor network.


DCP - Data Collection Protocol) - EnviSense
File Sensor Networks Window Help
DCP - Data Collection Proto
Bluetooth-Network
Base station
Clusterhead
Simple node
Bluetooth (Pool)
Base station
Clusterhead
Simple node
Chipcon (Dike)
Simple node
Configuration
Sensornode table
Topology
Routing
Node name | Software | Version | Vendor | MAC | Temperature | Pressure | Changed
Basis station | dcp | | Uni Rostock | 00-10-DC-71-92-67 | 25.2 | 1031 hPa | 10:20
Cluster 1 | dcp | V1.01 | Uni Rostock | 00-1d-9f-74-34-67 | 23.4 | 1025 hPa | 10:20
Cluster 2 | dcp | V1.01 | Uni Rostock | 00-6d-1D-41-64-52 | 24.7 | 1042 hPa | 10:20
Cluster 3 | dcp | V1.01 | Uni Rostock | 00-d0-22-71-92-67 | 25.2 | 1030 hPa | 10:20
Simple node 1 | dcp | V1.02 | Uni Rostock | 00-20-DC-71-a9-67 | 24.0 | 1027 hPa | 10:30
Simple node 2 | dcp | V1.02 | Uni Rostock | 00-1a-a1-81-23-12 | 25.1 | 1026 hPa | 10:30
Simple node 3 | dcp | V1.00 | Uni Rostock | 00-12-ab-26-92-67 | 24.7 | 1027 hPa | 10:30
Simple node 4 | dcp | V1.00 | Uni Rostock | 00-10-DC-df-26-75 | 23.9 | 1039 hPa | 10:30
Simple node 5 | dcp | V1.00 | Uni Rostock | 00-13-af-45-26-78 | 25.2 | 1023 hPa | 10:30
Simple node 6 | dcp | V1.02 | Uni Rostock | 00-60-57-d8-0b-5f | 24.6 | 1020 hPa | 10:30
Simple node 7 | dcp | V1.02 | Uni Rostock | 00-19-ae-71-92-62 | 25.0 | 1021 hPa | 10:30
Simple node 8 | dcp | V1.01 | Uni Rostock | 00-12-af-79-72-78 | 25.4 | 1038 hPa | 10:30
Simple node 9 | dcp | V1.01 | Uni Rostock | 00-31-74-dc-97-98 | 24.6 | 1035 hPa | 10:30
Welcome to EnviSense V0.02 Build 132 compiled on 2004/12/08.

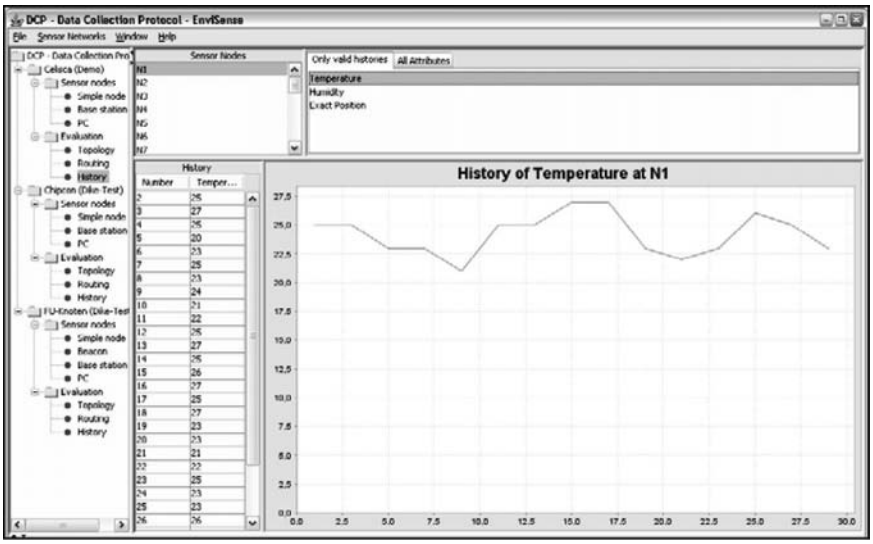


FIGURE 12.30 Screenshot of EnviSense with (a) attributes of connected nodes and (b) history of one attribute.

12.6 Summary

At the present time, TinyOS is the most mature OS framework for sensor nodes. The component-based architecture of TinyOS allows an easy composition of SNAs. New components can be added easily to TinyOS to support novel sensing or transmission technologies or to support upcoming sensor node platforms. MATÉ addresses the requirement to change a sensor node's behavior at runtime by introducing a VM on top of TinyOS. Via transmitting capsules containing high-level instructions, a wide range of SNAs can be installed dynamically into a deployed sensor network. TinyDB was developed to simplify data querying from sensor networks. On top of TinyOS, it provides an easy to use SQL interface to express data queries and addresses the group of users nonexperienced with writing embedded C code for sensor nodes. TOSSIM is a simulator for wireless sensor networks based on the TinyOS framework.

EnviroTrack is an object-based programming model to develop sensor network applications for tracking activities in the physical environment. Its main feature is dynamical grouping of nodes depending on environmental changes described by predefined aggregate functions, critical mass, and freshness horizon. SensorWare is a software framework for sensor networks employing lightweight and mobile control scripts that allow the dynamic deployment of distributed algorithms into a sensor network. In comparison to the MATÉ framework, the SensorWare runtime environment supports multiple applications to run concurrently on one SensorWare node. The MiLAN middleware provides a framework to optimize network performance, which needs sensing probability and energy costs based on equations. It is the programmer's decision to weight these equations.

EmStar is a software environment for developing and deploying applications for sensor networks consisting of 32 bit embedded Microserver platforms. SeNeTs is a new approach to optimize the interfaces of sensor network middleware. SeNeTs aims at the development of energy-saving applications and the resolving of component dependencies at compile time.

References

1. J. M. Kahn, R. H. Katz, and K. S. J. Pister, Next century challenges: Mobile networking for smart dust, in *Proc. of the ACM MobiCom'99*, Washington, 1999, pp. 271–278.
2. D. Culler, E. Brewer, and D. Wagner, A platform for WEbS (wireless embedded sensor actuator systems), Technical Report, University of California, Berkeley, CA, 2001.
3. G. J. Pottie and W. J. Kaiser, Wireless integrated network sensors, *Communications of the ACM*, 43(5), 51–58, 2000.
4. J. Rabaey, M. J. Ammer, J. L. da Silva, D. Patel, and S. Roundy, PicoRadio supports ad hoc ultra-low power wireless networking, *IEEE Computer*, 33(7), 42–48, July 2000.
5. EYES—Energy-efficient sensor networks, URL: http://eyes.eu.org
6. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, A survey on sensor networks, *IEEE Communications Magazine*, 40(8), 102–114, August 2002.
7. P. Rentala, R. Musunuri, S. Gandham, and U. Saxena, Survey on sensor networks, Technical Report UTDCS-10-03, University of Texas, Richardson, TX, 2003.
8. P. Levis and D. Culler, MATÉ: A tiny virtual machine for sensor networks, in *Proc. of ACM Conference on Architecture Support for Programming Languages and Operating Systems (ASPLOS)*, San Jose, CA, October 2002.
9. J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, System architecture directions for networked sensors, in *Proc. of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems*, Cambridge, MA, November 2000.
10. D. Gay, P. Levis, R. v. Behren, M. Welsh, E. Brewer, and D. Culler, The nesC language: A holistic approach to networked embedded systems, in *Proc. of Conference on Programming Language Design and Implementation (PLDI)*, San Diego, CA, June 2003.

11. S. Madden, J. Hellerstein, and W. Hong, TinyDB: In-network query processing in TinyOS, Intel Research, IRB-TR-02-014, Santa Clara, CA, October 2002.
12. A. Boulis and M. B. Srivastava, A framework for efficient and programmable sensor networks, in *Proc. of the Fifth IEEE Conference on Open Architectures and Network Programming (OPENARCH 2002)*, New York, June 2002.
13. A. Murphy and W. Heinzelman, MiLAN: Middleware linking applications and networks, Technical Report, University of Rochester, Computer Science Department, Rochester, New York, URL: http://hdl.handle.net/1802/305, January 2003.
14. M. Perillo and W. Heinzelman, Providing application QoS through intelligent sensor management, in *Proc. of the First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA'03)*, Anchorage, AK, May 2003.
15. T. Abdelzaher, B. Blum, et al., EnviroTrack: An environmental programming model for tracking applications in distributed sensor networks, Technical Report CS-2003-02, University of Virginia, Charlottesville, VA, 2003.
16. D. Culler, TinyOS—a component-based OS for the networked sensor regime, URL: http://webs.cs.berkeley.edu/tos/, 2003.
17. J. Blumenthal, M. Handy, F. Golatowski, M. Haase, and D. Timmermann, Wireless sensor networks—new challenges in software engineering, in *Proc. of the Ninth IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Lisbon, Portugal, September 2003.
18. A. Dunkels, B. Grönvall, and T. Voigt, Contiki—a lightweight and flexible operating system for tiny networked sensors, in *Proc. of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, Tampa, FL, 2004.
19. A. Dunkels, Contiki—a dynamic operating system for memory-constrained networked embedded systems, http://www.sics.se/contiki, 2008.
20. Open Geospatial Consortium, Inc., http://www.opengeospatial.org
21. The Network Simulator—ns-2, http://www.isi.edu/nsnam/ns
22. Berkeley WEBS: TinyOS, http://today.cs.berkeley.edu/tos/, 2004.
23. P. Levis, N. Lee, M. Welsh, and D. Culler, TOSSIM: Accurate and scalable simulation of entire TinyOS applications, in *Proc. of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, Los Angeles, November 2003.
24. TOSSIM: A simulator for TinyOS networks—user's manual, in TinyOS documentation.
25. L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin, EmStar: A software environment for developing and deploying wireless sensor networks, in *Proc. of USENIX 04*, Boston, MA, June 2004.
26. EmStar: Software for wireless sensor networks; URL: http://cvs.cens.ucla.edu/emstar/, 2004.
27. J. Blumenthal, M. Handy, and D. Timmermann, SeNeTs—test and validation environment for applications in large-scale wireless sensor networks, in *Proc. of the Second IEEE International Conference on Industrial Informatics INDIN'04*, Berlin, Germany, June 2004.
28. J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris, Capacity of ad hoc wireless networks, in *Proc. of MobiCom*, Rome, Italy, July 2001.
29. A. Sobeih, W.-P. Chen, J. C. Hou, L.-C. Kung, N. Li, H. Lim, H.-Y. Tyan, and H. Zhang, J-Sim: A simulation and emulation environment for wireless sensor networks, http://www.j-sim.org/v1.3/sensor/JSim.pdf, Ohio, 2004.
30. S. Park, A. Savvides, and M. B. Srivastava, SensorSim: A simulation framework for sensor networks, in *Proc. MSWIM 2000*, Boston, MA, 2000.
31. C. E. Perkins, E. M. Royer, and S. R. Das, Ad hoc on demand distance vector (AODV) routing, IETF Internet Draft, 2001.
32. B. Karpand and H. Kung, Greedy perimeter stateless routing for wireless networks, in *Proc. of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom2000)*, pp. 243–254, Boston, MA, 2000.

33. L. Bajaj, M. Takai, R. Ahuja, R. Bagrodia, and M. Gerla, GloMoSim: A scalable network simulation environment, Technical Report 990027, Computer Science Department, University of California Los Angeles, 1999.
34. OPNET: http://www.opnet.com
35. Moteworks User's Manual, Crossbow, May 2007.
36. W. Hong, Tiny application sensor kit (TASK), Intel Research, Berkeley, CA, 2003.
37. J. Blumenthal, F. Reichenbach, F. Golatowski, and D. Timmermann: Controlling wireless sensor networks using SeNeTs and EnviSense, in *Third IEEE International Conference on Industrial Informatics, INDIN 05*, ISBN: 0-7803-9095-4, Perth, Australia, 2005.

III

Automotive Networked Embedded Systems

13 Trends in Automotive Communication Systems *Nicolas Navet and Françoise Simonot-Lion* **13**-1
Automotive Communication Systems: Characteristics and Constraints • In-Car Embedded Networks • Middleware Layer • Open Issues for Automotive Communication Systems

14 Time-Triggered Communication *Roman Obermaisser* **14**-1
Time- and Event-Triggered Communication • Time-Triggered Communication in the Automotive Domain • Dependability Concepts • Fundamental Services of a Time-Triggered Communication Protocol • Time-Triggered Communication Protocols

15 Controller Area Networks for Embedded Systems *Gianluca Cena and Adriano Valenzano* **15**-1
Introduction • CAN Protocol Basics • Schedulability Analysis of CAN Networks • Considerations about CAN • Time-Triggered CAN • CANopen • CANopen Device Profile for Generic I/O Modules

16 FlexRay Communication Technology *Roman Nossal-Tueyeni and Dietmar Millinger* **16**-1
Introduction • Automotive Requirements • What is FlexRay? • System Configuration • Standard Software Components

17 LIN Standard *Antal Rajnak* **17**-1
Introduction • The Need • History • Some LIN Basics • LIN Physical Layer • LIN Protocol • Design Process and Workflow • System Definition Process • Debugging • Future • LIN Network Architect • LIN Target Package • LIN Spector—Test Tool • Summary • Acknowledgments

18 Standardized System Software for Automotive Applications *Thomas M. Galla* **18**-1
Introduction • Hardware Architecture • OSEK/VDX • HIS • ISO • ASAM • AUTOSAR • JasPar • Summary

19 Volcano: Enabling Correctness by Design *Antal Rajnak* **19**-1
Introduction • Volcano Concepts • Volcano Signals and the Publish/Subscribe Model • Update Bits • Flags • Timeouts • Frames • Network Interfaces • Volcano API • Volcano Resource Information • Execution Time of Volcano Processing Calls • Timing Model • Jitter • Capture of Timing Constraints • Volcano Network Architect • VNA: Tool Overview • Volcano Configuration • Configuration Files • Workflow • Acknowledgment

13

Trends in Automotive Communication Systems

13.1 Automotive Communication Systems: Characteristics and Constraints 13-1
From Point-to-Point to Multiplexed Communications • Car Domains and Their Evolution • Different Networks for Different Requirements • Event-Triggered versus TT
13.2 In-Car Embedded Networks 13-5
Priority Buses • Time-Triggered Networks • Low-Cost Automotive Networks • Multimedia Networks
13.3 Middleware Layer 13-14
13.4 Open Issues for Automotive Communication Systems 13-20
Optimized Networking Architectures • System Engineering
References 13-21

Nicolas Navet
INRIA—RealTime-at-Work

Françoise Simonot-Lion
LORIA—University of Nancy

13.1 Automotive Communication Systems: Characteristics and Constraints

13.1.1 From Point-to-Point to Multiplexed Communications

Since the 1970s, one observes an exponential increase in the number of electronic systems that have gradually replaced those that are purely mechanical or hydraulic. The growing performance and reliability of hardware components and the possibilities brought by software technologies enabled implementing complex functions that improve the comfort of the vehicle's occupants as well as their safety. In particular, one of the main purposes of electronic systems is to assist the driver to control the vehicle through functions related to the steering, traction (i.e., control of the driving torque), or braking such as the anti-lock braking system (ABS), electronic stability program (ESP), Electric Power Steering, active suspensions, or engine control. Another reason for using electronic systems is to control devices in the body of a vehicle such as lights, wipers, doors, windows, and, recently, entertainment and communication equipments (e.g., radio, DVD, hand-free phones, and navigation systems).

In the early days of automotive electronics, each new function was implemented as a stand-alone electronic control unit (ECU), which is a subsystem composed of a microcontroller and a set of sensors and actuators. This approach quickly proved to be insufficient with the need for functions to be distributed over several ECUs and the need for information exchanges among functions. For example, the vehicle speed estimated by the engine controller or by wheel rotation sensors has to be known to adapt the steering effort, to control the suspension, or simply to choose the right wiping speed.

In today's luxury cars, up to 2500 signals (i.e., elementary information such as the speed of the vehicle) are exchanged by up to 70 ECUs [1]. Until the beginning of the 1990s, data were exchanged through point-to-point links between ECUs. However this strategy, which required an amount of communication channels of the order of n^2 where n is the number of ECUs (i.e., if each node is interconnected with all the others, the number of links grows in the square of n), was unable to cope with the increasing use of ECUs due to the problems of weight, cost, complexity, and reliability induced by the wires and the connectors. These issues motivated the use of networks where the communications are multiplexed over a shared medium, which consequently required defining rules—protocols—for managing communications and, in particular, for granting bus access. It was mentioned in a 1998 press release (quoted in Ref. [33]) that the replacement of a "wiring harness with LANs in the four doors of a BMW reduced the weight by 15 kg." In the mid-1980s, the third part supplier Bosch developed controller area network (CAN), which was first integrated in Mercedes production cars in the early 1990s. Today, it has become the most widely used network in automotive systems and it is estimated [30] that the number of CAN nodes sold per year is currently around 400 millions (all application fields). Other communication networks, providing different services, are now being integrated in automotive applications. A description of the major networks is given in Section 13.2.

13.1.2 Car Domains and Their Evolution

As all the functions embedded in cars do not have the same performance or safety needs, different quality of services (QoS) (e.g., response time, jitter, bandwidth, redundant communication channels for tolerating transmission errors, efficiency of the error detection mechanisms, etc.) are expected from the communication systems. Typically, an in-car embedded system is divided into several functional domains that correspond to different features and constraints. Two of them are concerned specifically with real-time control and safety of the vehicle's behavior: the "powertrain" (i.e., control of engine and transmission) and the "chassis" (i.e., control of suspension, steering, and braking) domains. The third, the "body," mostly implements comfort functions. The "telematics" (i.e., integration of wireless communications, vehicle monitoring systems, and location devices), "multimedia," and "human machine interface" (HMI) domains take advantage of the continuous progress in the field of multimedia and mobile communications. Finally, an emerging domain is concerned with the safety of the occupant.

The main function of the powertrain domain is controlling the engine. It is realized through several complex control laws with sampling periods of a magnitude of some milliseconds (due to the rotation speed of the engine) and implemented in microcontrollers with high computing power. To cope with the diversity of critical tasks to be treated, multitasking is required and stringent time constraints are imposed on the scheduling of the tasks. Furthermore, frequent data exchanges with other car domains, such as the chassis (e.g., ESP, ABS) and the body (e.g., dashboard, climate control), are required.

The chassis domain gathers functions such as ABS, ESP, Automatic Stability Control, 4 Wheel Drive, which control the chassis components according to steering/braking solicitations and driving conditions (ground surface, wind, etc.). Communication requirements for this domain are quite similar to those for the powertrain but, because they have a stronger impact on the vehicle's stability, agility, and dynamics, the chassis functions are more critical from a safety standpoint. Furthermore, the "X-by-Wire" technology, currently used for avionic systems, is now slowly being introduced to execute steering or braking functions. X-by-Wire is a generic term referring to the replacement of mechanical or hydraulic systems by fully electrical/electronic ones, which led and still leads to new design methods for developing them safely [72] and, in particular, for mastering the interferences between functions [4]. Chassis and powertrain functions operate mainly as closed-loop control systems and their implementation is moving toward a time-triggered (TT) approach [32,48,53,60],

which facilitates composability (i.e., ability to integrate individually developed components) and deterministic real-time behavior of the system.

Dashboard, wipers, lights, doors, windows, seats, mirrors, and climate control are increasingly controlled by software-based systems that make up the body domain. This domain is characterized by numerous functions that necessitate many exchanges of small pieces of information among themselves. Not all nodes require a large bandwidth, such as the one offered by CAN; this leads to the design of low-cost networks such as local interconnect network (LIN) and TTP/A (see Section 13.2). On these networks, only one node, termed the master, possesses an accurate clock and drives the communication by polling the other nodes—the slaves—periodically. The mixture of different communication needs inside the body domain leads to a hierarchical network architecture where integrated mechatronic subsystems based on low-cost networks are interconnected through a CAN backbone. The activation of body functions is mainly triggered by the driver and passengers' solicitations (e.g., opening a window, locking doors, etc.).

Telematics functions are becoming more and more numerous: hand-free phones, car radio, CD, DVD, in-car navigation systems, rear seat entertainment, remote vehicle diagnostic, etc. These functions require a lot of data to be exchanged within the vehicle but also with the external world through the use of wireless technology (see, for instance, Ref. [58]). Here, the emphasis shifts from messages and tasks subject to stringent deadline constraints to multimedia data streams, bandwidth sharing, multimedia QoS where preserving the integrity (i.e., ensuring that information will not be accidentally or maliciously altered) and confidentiality of information is crucial. HMI aims to provide HMI that are easy to use and that limit the risk of driver inattention [17].

Electronic-based systems for ensuring the safety of the occupants are increasingly embedded in vehicles. Examples of such systems are impact and roll-over sensors, deployment of airbags and belt pretensioners, tyre pressure monitoring or Adaptive Cruise Control (the car's speed is adjusted to maintain a safe distance with the car ahead). These functions form an emerging domain usually referred to as "active and passive safety."

13.1.3 Different Networks for Different Requirements

The steadily increasing need for bandwidth* and the diversification of performance, costs, and dependability† requirements lead to a diversification of the networks used throughout the car. In 1994, the Society for Automotive Engineers (SAE) defined a classification for automotive communication protocols [11,64,65] based on data transmission speed and functions that are distributed over the network. "Class A" networks have a data rate lower than 10 kbit/s and are used to transmit simple control data with low-cost technology. They are mainly integrated in the body domain (seat control, door lock, lighting, trunk release, rain sensor, etc.). Examples of class A networks are LIN [35,56] and TTP/A [23]. "Class B" networks are dedicated to supporting data exchanges between ECUs to reduce the number of sensors by sharing information. They operate from 10 to 125 kbit/s. The J1850 [66] and low-speed CAN [25] are the main representations of this class. Applications that need high speed real-time communications require "class C" networks (speed of 125 kbit/s to 1 Mbit/s) or "class D" networks‡ (speed over 1 Mb/s). Class C networks, such as high-speed CAN [27], are used for the powertrain and currently for the chassis domains, while class D networks are devoted to multimedia data (e.g., media oriented system transport (MOST) [39]) and safety-critical applications

* For instance, in Ref. [4], the average bandwidth needed for the engine and the chassis control is estimated to reach 1500 kbit/s in 2008 while it was 765 kbit/s in 2004 and 122 kbit/s in 1994.

† Dependability is usually defined as the ability to deliver a service that can justifiably be trusted, see Ref. [3] for more details.

‡ Class D is not formally defined but it is generally considered that networks over 1 Mb/s belong to class D.

that need predictability and fault-tolerance (e.g., TTP/C [70] or FlexRay [10] networks) or serve as gateways between subsystems (see the use of FlexRay at BMW in Ref. [62]).

It is common, in today's vehicles, that the electronic architecture includes four different types of networks interconnected by gateways. For example, the Volvo XC90 [30] embeds up to 40 ECUs interconnected by a LIN bus, a MOST bus, a low-speed CAN, and a high-speed CAN. In the near future, it is possible that a bus dedicated to Occupant Safety Systems (e.g., airbag deployment, crash sensing) such as the "Safe-by-Wire plus" [7] will be added.

13.1.4 Event-Triggered versus TT

One of the main objectives of the design step of an in-vehicle embedded system is to ensure a proper execution of the vehicle functions, with a predefined level of safety, in the normal functioning mode but also when some components fail (e.g., reboot of an ECU) or when the environment of the vehicle creates perturbations (e.g., electromagnetic interferences [EMI] causing frames to be corrupted). Networks play a central role in maintaining the embedded systems in a "safe" state as most critical functions are now distributed and need to communicate. Thus, the different communication systems have to be analyzed in regard to this objective; in particular, messages transmitted on the bus must meet their real-time constraints, which mainly consist of bounded response times and bounded jitters.

There are two main paradigms for communications in automotive systems: TT and event-triggered. Event-triggered means that messages are transmitted to signal the occurrence of significant events (e.g., a door has been closed). In this case, the system possesses the ability to take into account, as quickly as possible, any asynchronous events such as an alarm. The communication protocol must define a policy to grant access to the bus to avoid collisions; for instance, the strategy used in CAN (see Section 13.2.1.1) is to assign a priority to each frame and to give the bus access to the highest priority frame. Event-triggered communication is very efficient in terms of bandwidth usage as only necessary messages are transmitted. Furthermore, the evolution of the system without redesigning existing nodes is generally possible, which is important in the automotive industry where incremental design is a usual practice. However, verifying that temporal constraints are met is not obvious and the detection of node failures is problematic.

When communications are TT, frames are transmitted at predetermined points in time, which is well-suited for the periodic transmission of messages as it is required in distributed control loops. Each frame is scheduled for transmission at one predefined interval of time, usually termed a slot, and the schedule repeats itself indefinitely. This medium access strategy is referred to as time division multiple access (TDMA). As the frame scheduling is statically defined, the temporal behavior is fully predictable; thus, it is easy to check whether the timing constraints expressed on data exchanges are met. Another interesting property of TT protocols is that missing messages are immediately identified; this can serve to detect, in a short and bounded amount of time, nodes that are presumably no longer operational. The first negative aspect is the inefficiency in terms of network utilization and response times with regard to the transmission of aperiodic messages (i.e., messages that are not transmitted in a periodic manner). A second drawback of TT protocols is the lack of flexibility even if different schedules (corresponding to different functioning modes of the application) can be defined and switching from one mode to another is possible at run-time. Finally, the unplanned addition of a new transmitting node on the network induces changes in the message schedule and, thus, necessitates the update of all other nodes. TTP/C [70] is a purely TT network but there are networks, such as time-triggered controller area network (TTCAN) [29], FTT-CAN [16] and FlexRay, that can support a combination of both TT and event-triggered transmissions. This capability to convey both types of traffic fits in well with the automotive context since data for control loops as well as alarms and events have to be transmitted.

Several comparisons have been done between event-triggered and TT approaches, the reader can refer to Refs. [1,16,31] for good starting points.

13.2 In-Car Embedded Networks

The different performance requirements throughout a vehicle, as well as competition among companies of the automotive industry, have led to the design of a large number of communication networks. The aim of this section is to give a description of the most representative networks for each main domain of utilization.

13.2.1 Priority Buses

To ensure at run-time the "freshness"* of the exchanged data and the timely delivery of commands to actuators, it is crucial that the Medium Access Control (MAC) protocol is able to ensure bounded response times of frames. An efficient and conceptually simple MAC scheme that possesses this capability is the granting of bus access according to the priority of the messages (the reader can refer to Refs. [12,43,68] for how to compute bound on response times for priority buses). To this end, each message is assigned an identifier, unique to the whole system. This serves two purposes: giving priority for transmission (the lower the numerical value, the greater the priority) and allowing message filtering upon reception. The two main representatives of such "priority buses" are CAN and J1850.

13.2.1.1 CAN Network

CAN is without a doubt the most widely used in-vehicle network. It was designed by Bosch in the mid-1980s for multiplexing communication between ECUs in vehicles and thus for decreasing the overall wire harness: length of wires and number of dedicated wires (e.g., the number of wires has been reduced by 40%, from 635 to 370, in the Peugeot 307 that embeds two CAN buses with regard to the nonmultiplexed Peugeot 306 [36]). Furthermore, it allows to share sensors among ECUs.

CAN on a twisted pair of copper wires became an ISO standard in 1994 [25,27] and is now a de-facto standard in Europe for data transmission in automotive applications, due to its low cost, its robustness, and the bounded communication delays (see Ref. [30]). In today's car, CAN is used as an SAE class C network for real-time control in the powertrain and chassis domains (at 250 or 500 kbit/s), but it also serves as an SAE class B network for the electronics in the body domain, usually at a data rate of 125 kbit/s.

On CAN, data, possibly segmented in several frames, may be transmitted periodically, aperiodically, or on-demand (i.e., client–server paradigm).

A CAN frame is labeled by an identifier, transmitted within the frame (see Figures 13.1 and 13.2), whose numerical value determines the frame priority. There are two versions of the CAN protocol differing in the size of the identifier: CAN 2.0A (or "standard CAN") with an 11 bit identifier and CAN 2.0B (or "extended CAN") with a 29 bit identifier. For in-vehicle communications, only CAN 2.0A is used as it provides a sufficient number of identifiers (i.e., the number of distinct frames exchanged over one CAN network is lower than 2^{11}).

CAN uses non-return-to-zero (NRZ) bit representation with a bit stuffing of length 5. In order not to lose the bit time (i.e., the time between the emission of two successive bits of the same frame), stations need to resynchronize periodically and this procedure requires edges on the signal. Bit stuffing is an encoding method that enables resynchronization when using NRZ bit representation where the signal level on the bus can remain constant over a longer period (e.g., transmission of "000000…").

* The freshness property is verified if data have been produced recently enough to be safely consumed: the difference between the time when data are used and the last production time must be always smaller than a specified value.

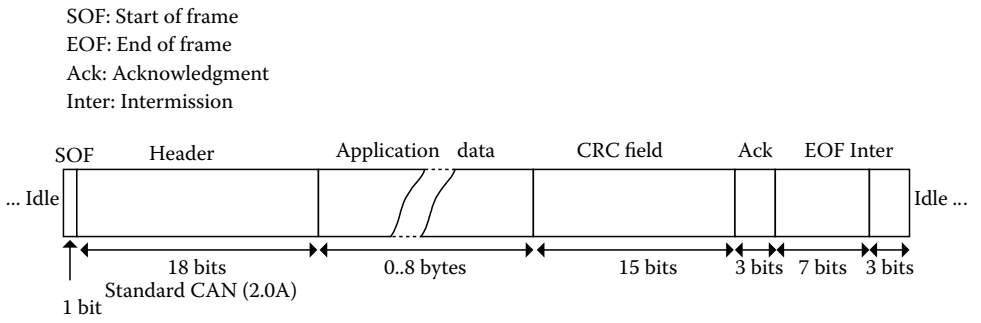


FIGURE 13.1 Format of the CAN 2.0A data frame. (From Navet, N., Song, Y.Q., Simonot-Lion, F., and Wilwert, C., *Proc. IEEE*., 93(6), 1204, 2005. With permission.)

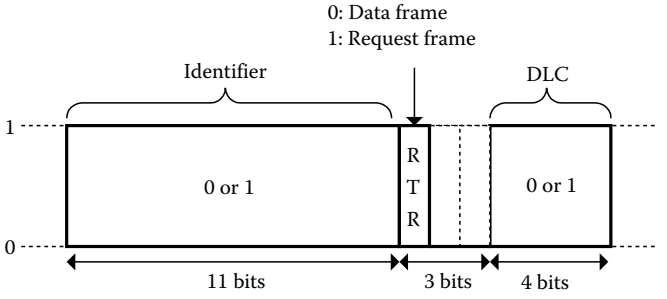


FIGURE 13.2 Format of the header field of the CAN 2.0A data frame. (From Navet, N., Song, Y.Q., Simonot-Lion, F., and Wilwert, C., *Proc. IEEE*., 93(6), 1204, 2005. With permission.)

Edges are generated into the outgoing bit stream in such a way to avoid the transmission of more than a maximum number of consecutive equal-level bits (5 for CAN). The receiver will apply the inverse procedure and de-stuff the frame. CAN requires the physical layer to implement the logical "and" operator: if at least one node is transmitting the "0" bit level on the bus, then the bus is in that state regardless if other nodes have transmitted the "1" bit level. For this reason, 0 is termed the dominant bit value while 1 is the recessive bit value.

The standard CAN data frame (CAN 2.0A, see Figure 13.1) can contain up to 8 bytes of data for an overall size of, at most, 135 bits, including all the protocol overheads such as the stuff bits. The sections of the frames are

- Header field (see Figure 13.2), which contains the identifier of the frame, the remote transmission request (RTR) bit that distinguishes between data frame (RTR set to 0) and data request frame (RTR set to 1) and the data length code (DLC) used to inform of the number of bytes of the data field.
- Data field having a maximum length of 8 bytes.
- 15 bit cyclic redundancy code (CRC) field which ensures the integrity of the data transmitted.
- Acknowledgment field (Ack). On CAN, the acknowledgment scheme solely enables the sender to know that at least one station, but not necessarily the intended recipient, has received the frame correctly.
- End-of-frame (EOF) field and the intermission frame space which are the minimum number of bits separating consecutive messages.

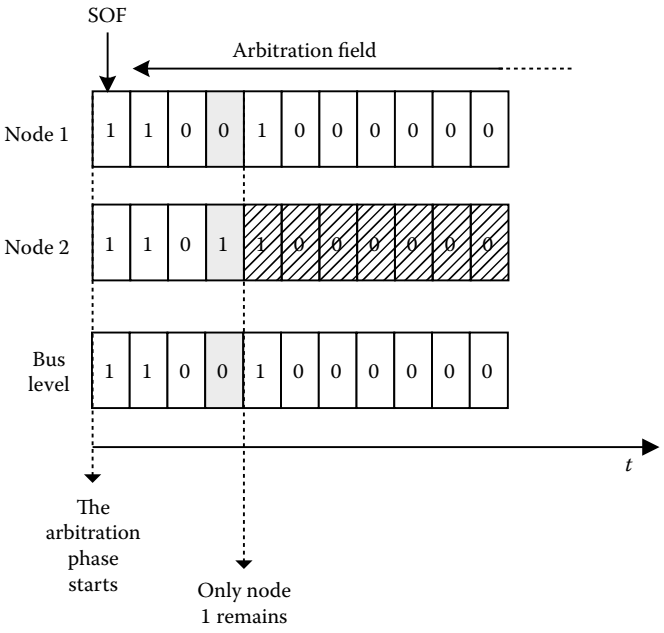

SOF
Arbitration field
Node 1
1 1 0 0 1 0 0 0 0 0 0
Node 2
1 1 0 1 1 0 0 0 0 0 0
Bus level
1 1 0 0 1 0 0 0 0 0 0
t
The arbitration phase starts
Only node 1 remains

FIGURE 13.3 CAN arbitration phase with two nodes starting transmitting simultaneously. Node 2 detects that a frame with a higher priority than its own is being transmitted when it monitors a level 0 (i.e., dominant level) on the bus while it has sent a bit with a level 1 (i.e., recessive level). Afterwards, Node 2 immediately stops transmitting. (From Navet, N., Song, Y.Q., Simonot-Lion, F., and Wilwert, C., *Proc. IEEE.*, 93(6), 1204, 2005. With permission.)

Any CAN node may start a transmission when the bus is idle. Possible conflicts are resolved by a priority-based arbitration process, which is said nondestructive in the sense that, in case of simultaneous transmissions, the highest priority frame will be sent despite the contention with lower priority frames. The arbitration is determined by the arbitration fields (identifier plus RTR bit) of the contending nodes. An example illustrating CAN arbitration is shown in Figure 13.3. If one node transmits a recessive bit on the bus while another transmits a dominant bit, the resulting bus level is dominant due to the "and" operator realized by the physical layer. Therefore, the node transmitting a recessive bit will observe a dominant bit on the bus and then will immediately stop transmitting. As the identifier is transmitted "most significant bit first," the node with the numerically lowest identifier field will gain access to the bus. A node that has lost the arbitration will wait until the bus becomes free again before trying to retransmit its frame. CAN arbitration procedure relies on the fact that a sending node monitors the bus while transmitting. The signal must be able to propagate to the most remote node and return back before the bit value is decided. This requires the bit time to be at least twice as long as the propagation delay which limits the data rate: for instance, 1 Mbit/s is feasible on a 40 m bus at maximum while 250 kbit/s can be achieved over 250 m. To alleviate the data rate limit, and extend the lifespan of CAN further, car manufacturers are starting to optimize the bandwidth usage by implementing "traffic shaping" strategies that are very beneficial in terms of response times (see, for instance, Ref. [22]).

CAN has several mechanisms for error detection. For instance, it is checked that the CRC transmitted in the frame is identical to the CRC computed at the receiver end, that the structure of the frame is valid, and that no bit-stuffing error occurred. Each station which detects an error sends an "error flag" which is a particular type of frame composed of six consecutive dominant bits that allow all the stations on the bus to be aware of the transmission error. The corrupted frame automatically reenters into the next arbitration phase, which might lead it to miss its deadline due to the additional delay.

The error recovery time, defined as the time from detecting an error until the possible start of a new frame, is 17–31 bit times. CAN possesses some fault-confinement mechanisms aimed at identifying permanent failures due to hardware dysfunctioning at the level of the microcontroller, communication controller, or physical layer. The scheme is based on error counters that are increased and decreased according to particular events (e.g., successful reception of a frame, reception of a corrupted frame, etc.). The relevance of the algorithms involved is questionable (see Ref. [19]) but the main drawback is that a node has to diagnose itself, which can lead to the nondetection of some critical errors. For instance, a faulty oscillator can cause a node to transmit continuously a dominant bit, which is one manifestation of the "babbling idiot" fault, see Ref. [51]. Furthermore, other faults such as the partitioning of the network into several subnetworks may prevent all nodes from communicating due to bad signal reflection at the extremities. Without additional fault-tolerance facilities, CAN is not suited for safety-critical applications such as some future X-by-Wire systems. For instance, a single node can perturb the functioning of the whole network by sending messages outside their specification (i.e., length and period of the frames). Many mechanisms were proposed for increasing the dependability of CAN-based networks (see Ref. [51] for an excellent survey), if each proposal solves a particular problem, they have not necessarily been conceived to be combined. Furthermore, the fault-hypotheses used in the design of theses mechanisms are not necessarily the same and the interactions between them remain to be studied in a formal way.

The CAN standard only defines the physical layer and Data Link layer (DLL). Several higher level protocols have been proposed, for instance, for standardizing startup procedures, implementing data segmentation or sending periodic messages (see OSEK/VDX and AUTOSAR in Section 13.3). Other higher-level protocols standardize the content of messages to ease the interoperability between ECUs. This is the case for J1939 which is used, for instance, in Scania's trucks and buses [71].

13.2.1.2 Vehicle Area Network

Vehicle area network (VAN) (see Ref. [26]) is very similar to CAN (e.g., frame format and data rate) but possesses some additional or different features that are advantageous from a technical point of view (e.g., no need for bit-stuffing, in-frame response: a node being asked for data answers in the same frame that contained the request). VAN was used for years in production cars by the French carmaker PSA Peugeot-Citroën in the body domain (e.g., for the 206 model) but, as it was not adopted by the market, it was abandoned in favor of CAN.

13.2.1.3 J1850 Network

The J1850 [66] is an SAE class B priority bus that was adopted in the United States for communications with nonstringent real-time requirements, such as the control of body electronics or diagnostics. Two variants of the J1850 are defined: a 10.4 kbit/s single-wire version and 41.6 kbit/s two-wire version. The trend in new designs seems to be the replacement of J1850 by CAN or a low-cost network such as LIN (see Section 13.2.3.1).

13.2.2 Time-Triggered Networks

Among communication networks, as discussed before, one distinguishes TT networks where activities are driven by the progress of time and event-triggered once where activities are driven by the occurrence of events. Both types of communication have advantages but one considers that, in general, dependability is much easier to ensure using a TT bus (refer, for instance, to Ref. [60] for a discussion on this topic). This explains that, currently, only TT communication systems are being considered for use in X-by-Wire applications. In this category, multiaccess protocols based on TDMA are particularly well suited; they provide deterministic access to the medium (the order of the transmissions is defined statically at the design time), and thus bounded response times. Moreover, their

regular message transmissions can be used as "heartbeats" for detecting the station failures. The three TDMA-based networks that could serve as gateways or for supporting safety-critical applications are TTP/C (see Ref. [70]), FlexRay (see Section 13.2.2.1) and TTCAN (see Section 13.2.2.2). FlexRay, which is backed by the world's automotive industry, is becoming the standard in the industry and is already used in the BMW X5 model since 2006 (see Ref. [62]). In the following, we choose not to discuss further TTP/C which, to the best of our knowledge, is no more considered for vehicles but is now used in aircraft electronic systems. However, the important experience gained over the years with TTP/C, in particular regarding fault-tolerance features (see Ref. [20]) and their formal validation (see Ref. [49]), will certainly be beneficial to FlexRay.

13.2.2.1 FlexRay Protocol

A consortium of major companies from the automotive field is currently developing the FlexRay protocol. The core members are BMW, Bosch, Daimler, General Motors, NXP Semiconductors, Freescale Semiconductor, and Volkswagen. The first publicly available specification of the FlexRay Protocol has been released in 2004, the current version of the specification [10] is available at http://www.flexray.com.

The FlexRay network is very flexible with regard to topology and transmission support redundancy. It can be configured as a bus, a star, or multi-star. It is not mandatory that each station possesses replicated channels nor a bus guardian, even though this should be the case for critical functions such as the Steer-by-Wire. At the MAC level, FlexRay defines a communication cycle as the concatenation of a TT (or static) window and an event-triggered (or dynamic) window. In each communication window, size of which is set statically at design time, two distinct protocols are applied. The communication cycles are executed periodically. The TT window uses a TDMA MAC protocol; the main difference with TTP/C is that a station in FlexRay might possess several slots in the TT window, but the size of all the slots is identical (see Figure 13.4). In the event-triggered part of the communication cycle, the protocol is Flexible Time Division Multiple Access (FTDMA): the time is divided into so-called mini-slots, each station possesses* a given number of mini-slots (not necessarily consecutive) and it can start the transmission of a frame inside each of its own mini-slots. A mini-slot remains idle if the station has nothing to transmit which actually induces a loss of bandwidth (see Ref. [9] for a discussion on that topic). An example of a dynamic window is shown in Figure 13.5: on channel B, frames have been transmitted in mini-slots n and $n + 2$ while mini-slot $n + 1$ has not been used. It is noteworthy that frame $n + 4$ is not received simultaneously on channels A and B as, in the dynamic window, transmissions are independent in both channels.

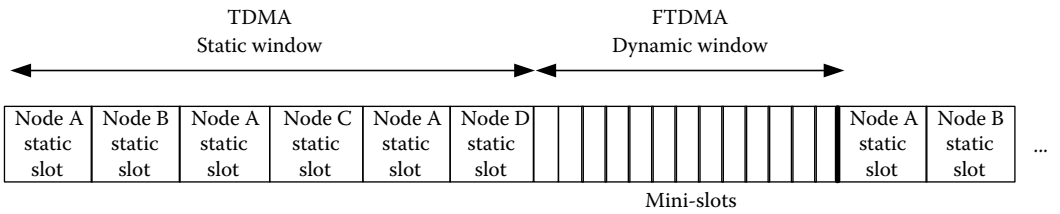


FIGURE 13.4 Example of a FlexRay communication cycle with 4 nodes A, B, C, and D. (From Navet, N., Song, Y.Q., Simonot-Lion, F., and Wilwert, C., *Proc. IEEE.*, 93(6), 1204, 2005. With permission.)

* Different nodes can send frames in the same slot but in different cycles, this is called "slot multiplexing" and it is only possible only in the dynamic segment.

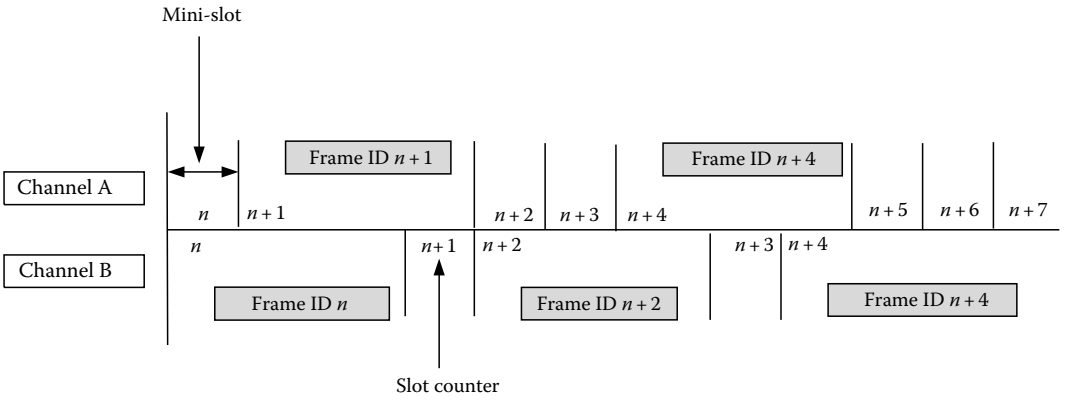


FIGURE 13.5 Example of message scheduling in the dynamic segment of the FlexRay communication cycle. (From Navet, N., Song, Y.Q., Simonot-Lion, F., and Wilwert, C., *Proc. IEEE.*, 93(6), 1204, 2005. With permission.)

The FlexRay MAC protocol is more flexible than the TTP/C MAC as in the static window nodes are assigned as many slots as necessary (up to 2047 overall) and as the frames are only transmitted if necessary in the dynamic part of the communication cycle. In a similar way as with TTP/C, the structure of the communication cycle is statically stored in the nodes, however, unlike TTP/C, mode changes with a different communication schedule for each mode are not possible.

The FlexRay frame consists of three parts: the header, the payload segment containing up to 254 bytes of data, and the CRC of 24 bits. The header of 5 bytes includes the identifier of the frame and the length of the data payload. The use of identifiers allows to move a software component, which sends a frame *X*, from one ECU to another ECU without changing anything in the nodes that consume frame *X*. It has to be noted that this is no more possible when signals produced by distinct components are packed into the same frame for the purpose of saving bandwidth (i.e., which is referred to as frame-packing or PDU-multiplexing—see Ref. [61] for this problem addressed on CAN).

From the dependability point of view, the FlexRay standard specifies solely the bus guardian and the clock synchronization algorithms. Other features, such as mode management facilities or a membership service, will have to be implemented in software or hardware layers on top of FlexRay (see, for instance, Ref. [5] for a membership service protocol that could be used along with FlexRay). This will allow to conceive and implement exactly the services that are needed with the drawback that correct and efficient implementations might be more difficult to achieve in a layer above the communication controller.

In the FlexRay specification, it is argued that the protocol provides scalable dependability, i.e., the "ability to operate in configurations that provide various degrees of fault tolerance." Indeed, the protocol allows for mixing links with single and dual transmission supports on the same network, subnetworks of nodes without bus-guardians or with different fault-tolerance capability with regard to clock synchronization, etc. In the automotive context where critical and noncritical functions will increasingly coexist and interoperate, this flexibility can prove to be efficient in terms of cost and reuse of existing components if missing fault-tolerance features are provided in a middleware (MW) layer, for instance such as the one currently under development within the automotive industry project AUTOSAR (see Section 13.3). The reader interested in more information about FlexRay can refer to Ref. [63], and to Refs. [21,54] for how to configure the communication cycle.

13.2.2.2 TTCAN Protocol

TTCAN (see Ref. [29]) is a communication protocol developed by Robert Bosch GmbH on top of the CAN physical and DLLs. TTCAN uses the CAN standard but, in addition, requires that the

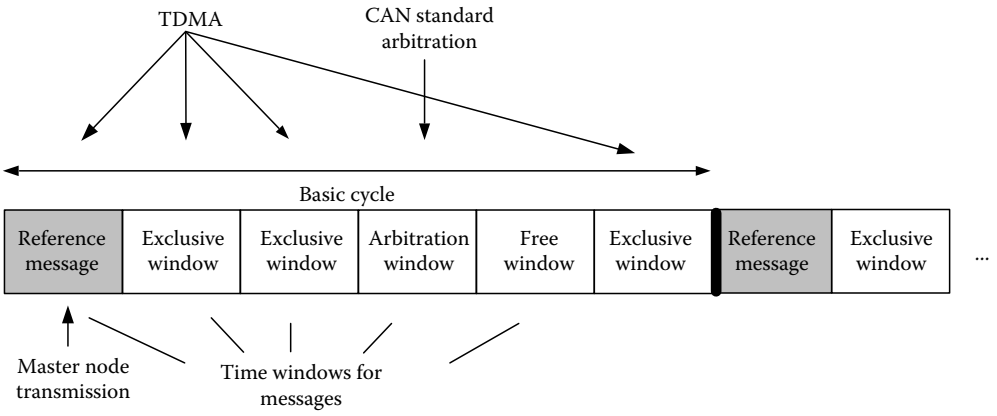


FIGURE 13.6 Example of a TTCAN basic cycle. (From Navet, N., Song, Y.Q., Simonot-Lion, F., and Wilwert, C., *Proc. IEEE.*, 93(6), 1204, 2005. With permission.)

controllers have the possibility to disable automatic retransmission of frames upon transmission errors and to provide the upper layers with the point in time at which the first bit of a frame was sent or received [59]. The bus topology of the network, the characteristics of the transmission support, the frame format, and the maximum data rate—1 Mbit/s—are imposed by CAN protocol. Channel redundancy is possible (see Ref. [38] for a proposal), but not standardized and no bus guardian is implemented in the node. The key idea is to propose, as with FlexRay, a flexible TT/event-triggered protocol. As illustrated in Figure 13.6, TTCAN defines a basic cycle (the equivalent of the FlexRay communication cycle) as the concatenation of one or several TT (or "exclusive") windows and one event-triggered (or "arbitrating") window. Exclusive windows are devoted to TT transmissions (i.e., periodic messages) while the arbitrating window is ruled by the standard CAN protocol: transmissions are dynamic and bus access is granted according to the priority of the frames. Several basic cycles, that differ by their organization in exclusive and arbitrating windows and by the messages sent inside exclusive windows, can be defined. The list of successive basic cycles is called the system matrix, which is executed in loops. Interestingly, the protocol enables the master node (i.e., the node that initiates the basic cycle through the transmission of the "reference message") to stop functioning in TTCAN mode and to resume in standard CAN mode. Later, the master node can switch back to TTCAN mode by sending a reference message.

TTCAN is built on a well-mastered and low-cost technology, CAN, but, as defined by the standard, does not provide important dependability services such as the bus guardian, membership service, and reliable acknowledgment. It is, of course, possible to implement some of these mechanisms at the application or MW level but with reduced efficiency. Some years ago, it was thought that carmakers could be interested in using TTCAN during a transition period until FlexRay technology is fully mature but this was not really the case and it seems that the future of TTCAN in production cars is rather unsure.

13.2.3 Low-Cost Automotive Networks

Several fieldbus networks have been developed to fulfill the need for low-speed/low-cost communication inside mechatronic-based subsystems generally made of an ECU and its set of sensors and actuators. Two representatives of such networks are LIN and TTP/A. The low-cost objective is achieved not only because of the simplicity of the communication controllers, but also because the requirements set on the microcontrollers driving the communication are reduced (i.e., low

computational power, small amount of memory, low-cost oscillator). Typical applications involving these networks include controlling doors (e.g., door locks, opening/closing windows) or controlling seats (e.g., seat position motors, occupancy control). Besides cost considerations, a hierarchical communication architecture, including a backbone such as CAN and several subnetworks such as LIN, enables reducing the total traffic load on the backbone.

Both LIN and TTP/A are master–slave networks where a single master node, the only node that has to possess a precise and stable time base, coordinates the communication on the bus: a slave is only allowed to send a message when it is polled. More precisely, the dialog begins with the transmission by the master of a "command frame" that contains the identifier of the message whose transmission is requested. The command frame is then followed by a "data frame" that contains the requested message sent by one of the slaves or by the master itself (i.e., the message can be produced by the master).

13.2.3.1 LIN Network

LIN (see Refs. [35,56]) is a low-cost serial communication system used as SAE class A network, where the needs in terms of communication do not require the implementation of higher-bandwidth multiplexing networks such as CAN. LIN is developed by a set of major companies from the automotive industry (e.g., DaimlerChrysler, Volkswagen, BMW, and Volvo) and is already widely used in production cars.

The LIN specification package (LIN version 2.1 [35]) includes not only the specification of the transmission protocol (physical and DLLs) for master–slave communications, but also the specification of a diagnostic protocol on top of the DLL. A language for describing the capability of a node (e.g., bit-rates that can be used, characteristics of the frames published, and subscribed by the node, etc.) and for describing the whole network is provided (e.g., nodes on the network, table of the transmissions' schedule, etc.). These description language facilitates the automatic generation of the network configuration by software tools.

A LIN cluster consists of one "master" node and several "slave" nodes connected to a common bus. For achieving a low-cost implementation, the physical layer is defined as a single wire with a data rate limited to 20 kbit/s due to EMI limitations. The master node decides when and which frame shall be transmitted according to the schedule table. The schedule table is a key element in LIN; it contains the list of frames that are to be sent and their associated frame-slots thus ensuring determinism in the transmission order. At the moment a frame is scheduled for transmission, the master sends a header (a kind of transmission request or command frame) inviting a slave node to send its data in response. Any node interested can read a data frame transmitted on the bus. As in CAN, each message has to be identified: 64 distinct message identifiers are available. Figure 13.7 depicts the LIN frame format and the time, termed a "frame-slot," during which a frame is transmitted.

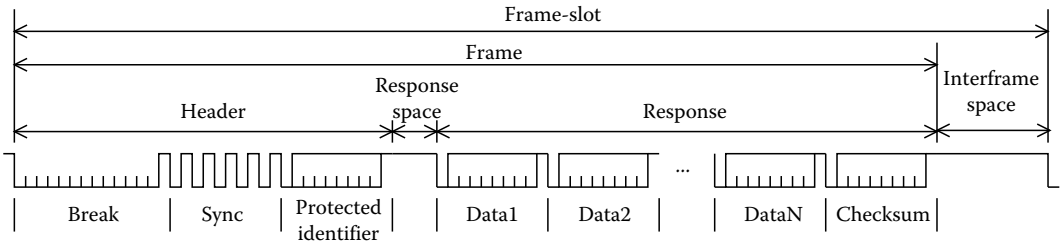


FIGURE 13.7 Format of the LIN frame. A frame is transmitted during its frame-slot, which corresponds to an entry of the schedule table. (From Navet, N., Song, Y.Q., Simonot-Lion, F., and Wilwert, C., *Proc. IEEE.*, 93(6), 1204, 2005. With permission.)

The header of the frame that contains an identifier is broadcast by the master node and the slave node that possesses this identifier inserts the data in the response field. The "break" symbol is used to signal the beginning of a frame. It contains at least 13 dominant bits (logical value 0) followed by one recessive bit (logical value 1) as a break delimiter. The rest of the frame is made of byte fields delimited by one start bit (value 0) and one stop bit (value 1), thus resulting in a 10-bit stream per byte. The "sync" byte has a fixed value (which corresponds to a bit stream of alternatively 0 and 1), it allows slave nodes to detect the beginning of a new frame and be synchronized at the start of the identifier field. The so-called "protected identifier" is composed of two subfields: the first 6 bits are used to encode the identifier and the last 2 bits, the identifier parity. The data field can contain up to 8 bytes of data. A checksum is calculated over the protected identifier and the data field. Parity bits and checksum enable the receiver of a frame to detect the bits that have been inverted during transmission.

LIN defines five different frame types: unconditional, event-triggered, sporadic, diagnostic, and user-defined. Frames of the latter type are assigned a specific identifier value and are intended to be used in an application-specific way that is not described in the specification. The first three types of frames are used to convey signals. Unconditional frames are the usual type of frames used in the master–slave dialog and are always sent in their frame-slots. Sporadic frames are frames sent by the master, only if at least one signal composing the frame has been updated. Usually, multiple sporadic frames are assigned to the same frame-slot and the higher priority frame that has an updated signal is transmitted. An event-triggered frame is used by the master willing to obtain a list of several signals from different nodes. A slave will only answer the master if the signals it produces have been updated, thus resulting in bandwidth savings if updates do not take place very often. If more than one slave answers, a collision will occur. The master resolves the collision by requesting all signals in the list one by one. A typical example of the use of the event-triggered transfer given in Ref. [34] is the doors' knob monitoring in a central locking system. As it is rare that multiple passengers simultaneously press a knob, instead of polling each of the four doors, a single event-triggered frame can be used. Of course, in the rare event when more than one slave responds, a collision will occur. The master will then resolve the collision by sending one by one the individual identifiers of the list during the successive frame-slots reserved for polling the list. Finally, diagnostic frames have a fixed size of 8 bytes, fixed value identifiers for both the master's request and the slave answers and always contain diagnostic or configuration data whose interpretation is defined in the specification.

It is also worth noting that LIN offers services to send nodes into a sleep mode (through a special diagnostic frame termed "go-to-sleep-command") and to wake them up, which is convenient as optimizing energy consumption, especially when the engine is not running, is a real matter of concern in the automotive context.

13.2.3.2 TTP/A Network

Like TTP/C, TTP/A [23] was initially invented at the Vienna University of Technology. TTP/A pursues the same aims and shares the main design principles as LIN and it offers, at the communication controller level, some similar functionalities, in particular, in the areas of plug-and-play capabilities and online diagnostics services. TTP/A implements the classic master–slave dialog, termed "master–slave round," where the slave answers the master's request with a data frame having a fixed length data payload of 4 bytes. The "Multipartner" rounds enable several slaves to send up to an overall amount of 62 bytes of data after a single command frame. A "broadcast round" is a special master–slave round in which the slaves do not send data; it is, for instance, used to implement sleep–wake-up services. The data rate on a single-wire transmission support is, as for LIN, equal to 20 kbit/s, but other transmission supports enabling higher data rates are possible. To our best knowledge, TTP/A is not currently in use in production cars.

13.2.4 Multimedia Networks

Many protocols have been adapted or specifically conceived for transmitting the large amount of data needed by emerging multimedia applications in automotive systems. Two prominent protocols in this category are MOST and IDB-1394.

13.2.4.1 MOST Network

MOST (see Ref. [39]) is a multimedia network development, which was initiated in 1998 by the MOST Cooperation (a consortium of carmakers and component suppliers). MOST provides point-to-point audio and video data transfer with different possible data rates. This supports end-user applications like radios, global positioning system (GPS) navigation, video displays, and entertainment systems. MOST's physical layer is a plastic optical fiber (POF) transmission support, which provides a much better resilience to EMI and higher transmission rates than classical copper wires. Current production cars, around 50 model series according to Ref. [37], for instance from BMW and Daimler, employ a MOST network, which is now becoming the de-facto standard for transporting audio and video within vehicles (see Refs. [40,41]). At the time of writing, the third revision of MOST has been announced with, as a new feature, the support of a channel that can transport standard Ethernet frames.

13.2.4.2 IDB-1394 Network

IDB-1394 is an automotive version of IEEE-1394 for in-vehicle multimedia and telematic applications jointly developed by the IDB Forum (see http://www.idbforum.org) and the 1394 Trade Association (see http://www.1394ta.org). The system architecture of IDB-1394 permits existing IEEE-1394 consumer electronics devices to interoperate with embedded automotive grade devices. IDB-1394 supports a data rate of 100 Mbps over twisted pair or POF, with a maximum number of embedded devices which are limited to 63 nodes. From the point of view of transmission rate and interoperability with existing IEEE-1394 consumer electronic devices, IDB-1394 was at some time considered a serious competitor for MOST technology but, despite a few early implementations at Renault and Nissan, as far as we know the protocol did not reach wide acceptance on the market.

13.3 Middleware Layer

The design of automotive electronic systems has to take into account several constraints. First, nowadays, the performance, quality, and safety of a vehicle depend on functions that are mainly implemented in software (e.g., the ignition control, ABS, wipper control, etc.) and moreover depend on a tight cooperation between these functions; e.g., the control of the engine is done according to requests from the driver (speeding up, slowing down as transmitted by the throttle position sensor or the brake pedal) and requirements from other embedded functions such as climate control or ESP. Second, in-vehicle embedded systems are produced through a complex cooperative multipartner development process shared between OEMs and suppliers. Therefore, to increase the efficiency of the production of components and their integration, two important problems have to be solved: (1) the portability of components from one ECU to another one enabling some flexibility in the architecture design, and (2) the reuse of components between platforms which is a keypoint especially for ECU suppliers. So, the cooperative development process raises the problem of interoperability of components. A classic approach for easing the integration of software components is to implement a "MW layer" that provides application programs with common services and a common interface. In particular, the common interface allows the design of an application disregarding the hardware platform and the distribution, and therefore enables the designer focusing on the development and the validation of the software components and the software architecture that realize a function.

Among the set of common services usually provided by an MW, those that related to the communication between several application components are crucial. They have to meet several objectives:

- *Hide the distribution* through the availability of services and interfaces that are the same for intra-ECU, inter-ECU, inter-domain communications whatever the underlying protocols.
- *Hide the heterogeneity* of the platform by providing an interface independent of the underlying protocols, of the CPU architecture (e.g., 8/16/32 bits, Endianness), of the operating systems (OS), etc.
- *Provide high-level services* to shorten the development time and increase quality through the reuse of validated services (e.g., working mode management, redundancy management, membership service, etc.). A good example of such a function is the "frame-packing" (sometimes also called "signal multiplexing") that enables application components to exchange "signals" (e.g., the number of revolutions per minute, the speed of the vehicle, the state of a light, etc.) while, at run-time, "frames" are transmitted over the network; so, the frame-packing service of an MW consists in packing the signals into frames and sending the frames at the right points in time for ensuring the deadline constraint on each signal it contains.
- *Ensure QoS properties required by the application*, in particular, it can be necessary to improve the QoS provided by the lower-level protocols as, e.g., by furnishing an additional CRC, transparent to the application, if the Hamming distance of the CRC specified by the network protocol is not sufficient in regard to the dependability objectives. Other examples are the correction of "bugs" in lower level protocols such as the "inconsistent message duplicate" of CAN (see Ref. [52]), the provision of a reliable acknowledgment service on CAN, the status information on the data consumed by the application components (e.g., data were refreshed since last reading, its freshness constraint was not respected, etc.) or filtering mechanisms (e.g., notify the application for each k reception or when the data value has changed in a significant way).

Note that a more advanced features would be to come up with adaptive communication services, thanks to algorithms that would modify at run-time the parameters of the communication protocols (e.g., priorities, transmission frequencies, etc.) according to the current requirements of the application (e.g., inner-city driving or highway driving) or changing environmental conditions (e.g., EMI level). For the time being, to the best of our knowledge, not such feature exists in automotive embedded systems. In fact, this point requires a coordinated approach for the design of function (as the definition of control law parameters, the identification of the parameters acting on the robustness of the function, etc.) and the deployment of the software architecture that implements the function (specifically the communication parameters). By increasing the efficiency and the robustness of the application, such an adaptive strategy would certainly ease the reusability.

Some proprietary MW were developed by several carmakers to support the integration of ECUs and software modules provided by their third-party suppliers. For instance, the TITUS/DBKOM communication stack is a proprietary MW of Daimler that standardizes the cooperation between components according to a client–server model. "Volcano" [8,55,57] is a commercial product of Mentor Graphics, initially developed in partnership with Volvo. The Volcano Target Package consists of a communication layer and a set of off-line configuration tools for application distributed on CAN and/or LIN. It is aimed to provide the mapping of signals into frames under network bandwidth optimization and ensure a predictable and deterministic real-time communication system thanks to schedulability analysis techniques (see Refs. [8,68]). To the best of our knowledge, no publicly available technically precise description of TITUS and Volcano exists.

A first step to define a standard for in-car embedded MW was started by the OSEK/VDX consortium (http://www.osek-vdx.org) . In particular, two specifications are of particular interest in the context of this chapter: the "OSEK/VDX Communication" layer [46] and the "Fault-Tolerant Communication (FTCom) layer" [45]. The first one specifies a communication layer [46] that defines common software interfaces and common behavior for internal and external communications between application components. How signals are packed into a frame is statically defined off-line and the OSEK/VDX Communication layer automatically realizes the packing/unpacking at run-time as well as the handling of queued or unqueued messages at the receiver side. OSEK/VDX Communication runs on top of a transport layer (e.g., Ref. [28]) that takes care mainly of possible segmentation of frames and it can operate on any OS compliant with "OSEK/VDX OS" services for tasks, events, and interrupt management (see Ref. [47]). Some questions deserve to be raised. In particular, communications between application processes that are internal to one ECU or located in two distant ECUs do not obey exactly the same rules (see Ref. [14] for more details); thus, the designer has to take into account the distribution of the functions, which is a hindrance to portability. Finally, OSEK/VDX Communication does not obey fully to a TT approach and is not intended to be used on top of a TT network, as for example TTP/C or FlexRay. For this purpose, "OSEK/VDX FTCom" (see Ref. [45]) is a proposal whose main functions is to manage the redundancy of data needed for achieving fault-tolerance (i.e., the same information can be produced by a set of replicated nodes) by presenting only one copy of data to the receiver application according to the agreement strategy specified by the designer. Two other important services of the OSEK/VDX FTCom, already offered by OSEK/VDX Communication, are (1) to manage the packing/unpacking of messages [61], and (2) to provide message filtering mechanisms for passing only "significant" data to the application. OSEK/VDX FTCom was developed to run on top of a TT OS such as OSEK Time [44].

Between 2001 and 2004, a European cooperative project aimed at the specification of an automotive MW within the automotive industry (ITEA EAST-EEA project—see http://www.east-eea.net) was undertaken. To the best of our knowledge, the ITEA EAST-EEA project was the first important initiative targeting the specification of both the services to be ensured by the MW and the architecture of the MW itself in terms of components and architecture of components. Similar objectives guide the work done in the AUTOSAR consortium, see http://www.autosar.org and [15,18], that gathers most the key players in the automotive industry. The specifications produced by the consortium become quickly de-facto standards for the cooperative development of in-vehicle embedded systems (see, for instance, the migration to AUTOSAR at PSA Peugeot-Citröen [13]).

AUTOSAR specifies the software architecture embedded in an ECU. More precisely, it provides a reference model which is comprised of three main parts:

- Application layer
- Basic software (MW software components)
- Run Time environment (RTE) that provides standardized software interfaces to the application software

One of AUTOSAR's main objective is to improve the quality and the reliability of embedded systems. By using a well-suited abstraction, the reference model supports the separation between software and hardware, it eases the mastering of the complexity, allows the portability of application software components and therefore the flexibility for product modification, upgrade and update, as well as the scalability of solutions within and across product lines. The AUTOSAR reference architecture is schematically illustrated in Figure 13.8. An important issue is the automatic generation of an AUTOSAR MW that has to be done from the basic software components, generally provided by suppliers, and the specification of the application itself (description of applicative-level tasks, signals sent or received, events, alarms, etc.). The challenge is to realize such a generation so that the deployment of the MW layer can be optimized for each ECU.

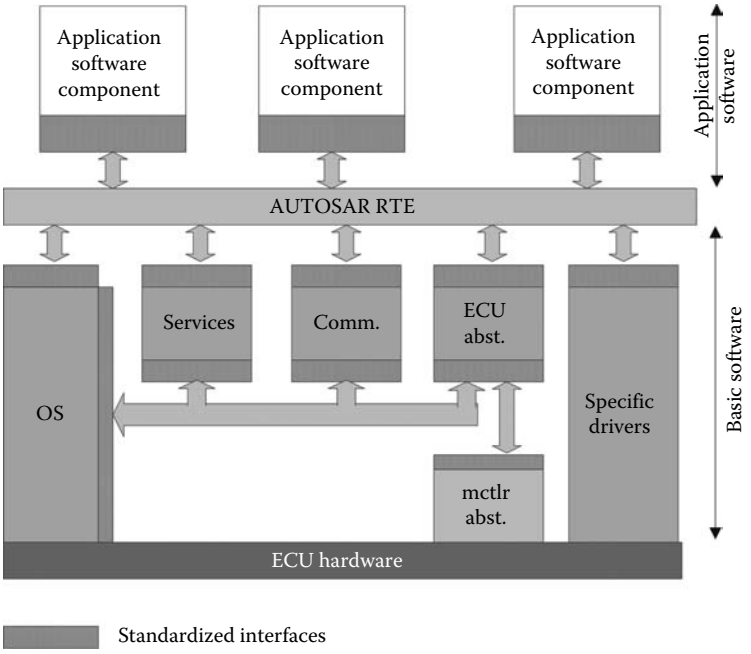


FIGURE 13.8 AUTOSAR reference architecture.

One of the main objectives of the AUTOSAR MW is to hide the characteristic of the hardware platform as well as the distribution of the application software components. Thus, the inter- or intra-ECU communication services are of major importance and are thoroughly described in the documents provided by the AUTOSAR consortium (see Figure 13.9 for an overview of the different modules). The role of these services is crucial for the behavioral and temporal properties of an embedded and distributed application. So, their design, their generation, and configuration have to be precisely mastered and the verification of timing properties becomes an important activity. The problem is complex because, as for the formerly mentioned MW the objects (e.g., signals, frames, I-PDU, etc.) that are handled by services at one level are not the same objects that are handled by services at another level. Nevertheless each object is strongly dependent of one or several objects handled by services belonging to neighboring levels. The AUTOSAR standard proposes two communication models:

- "Sender–receiver" used for passing information between two application software components (belonging to the same task, to two distinct tasks on the same ECU or to two remote tasks)
- "Client–server" that supports function invocation

Two communication modes are supported for the sender–receiver communication model:

- "Explicit" mode is specified by a component that makes explicit calls to the AUTOSAR MW for sending or receiving data.
- "Implicit" mode means that the reading (resp. writing) of data is automatically done by the MW before the invocation (resp. after the end of execution) of a component consuming (resp. producing) the data without any explicit call to AUTOSAR services; this is away to protect effectively the data between application software components and MW services.

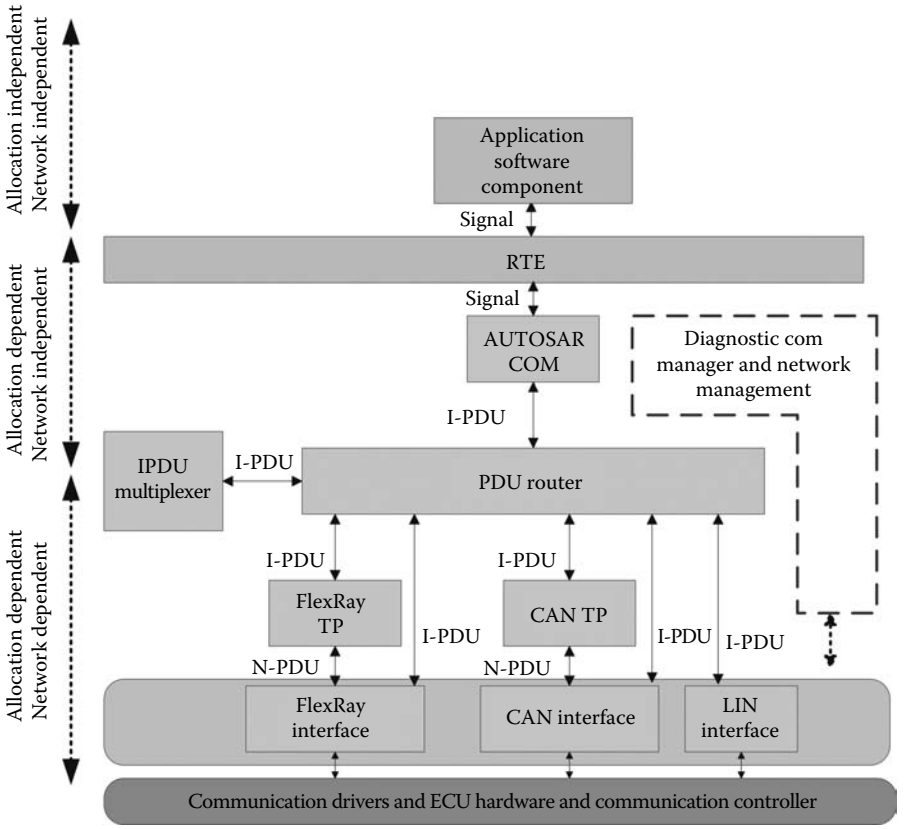


FIGURE 13.9 Communication software components and architecture.

AUTOSAR identifies three main objects regarding the communication: signal exchanged between software components at application level, I-PDU (Interaction Layer Protocol Data Unit) that consists of a group of one or several signals, and the N-PDU (Data Link Layer Protocol Data Unit) that will actually be transmitted on the network. Precisely AUTOSAR defines:

- Signals at application level that are specified by a length and a type. Conceptually a signal is exchanged between application software components through ports disregarding the distribution of this component. The application needs to precise a "Transfer Property" parameter that will impact the behavior of the transmission and whose value can be "triggered" (each time the signal is provided to the MW by the application, it has to be transmitted on the network) or "pending" (the actual transmission of a signal on the network depends only on the emission rule of the frame that contains the signal). Furthermore, when specifying a signal, the designer has to indicate if it is a "data" or an "event." In the former case, incoming data are not queued on the receiver side while in the latter one, signals are queued on the receiver side and therefore, for each transmission of the signal, a new value will be made available to the application. The handling of buffers or queues is done by the RTE.
- I-PDU are built by the AUTOSAR COM component. Each I-PDU is made of one or several signals and is passed via the PDU Router to the communication interfaces. The maximum length of an I-PDU depends on the maximum length of the L-PDU (i.e., DLL PDU) of the underlying communication interface: for CAN and LIN the maximum

L-PDU length is 8 bytes while for FlexRay the maximum L-PDU length is 254 bytes. AUTOSAR COM ensures a local transmission when both components are located on the same ECU, or by building suited objects and triggering the appropriate services of the lower layers when the components are remote. This scheme enables the portability of components and hides their distribution. The transformation from signals to I-PDU and from I-PDU to signals is done according to an offline generated configuration. Each I-PDU is characterized by a behavioral parameter, termed "Transmission Mode" whose possible value is "direct" indicates (the sending of the I-PDU is done as soon as a triggered signal contained in this I-PDU is sent at application layer), "periodic" means (the sending of the I-PDU is done only periodically), "mixed" (the rules imposed by the triggered signals contained in the I-PDU are taken into account, and additionally the I-PDU is sent periodically if it contains at least one pending signal) or "none" (for I-PDUs whose emission rules depend on the underlying network protocol, as e.g., FlexRay; no transmission is initiated by AUTOSAR COM in this mode).

- N-PDU is built by the basic components CAN TP (Transport Protocol) or FlexRay TP. It consists of the data payload of the frame that will be transmitted on the network and protocol control information. Note that the use of a transport layer is not mandatory and I-PDUs can be transmitted directly to the lower layers (see Figure 13.9).

The RTE implements the AUTOSAR MW interface and the corresponding services. In particular, the RTE handles the "implicit/explicit" communication modes and the fact that the communication involves "events" (queued) or "data" (unqueued). The AUTOSAR COM component is responsible for several functions: on the sender side, it ensures the transmission and notifies the application about its outcome (success or error). In particular, AUTOSAR COM can inform the application if the transmission of an I-PDU did not take place before a specified deadline (i.e., deadline monitoring). On the receiver side, it also notifies the application (success or error of a reception) and supports the filtering mechanism for signals (dispatching each signal of a received I-PDU to the application or to a gateway). Both at the sending and receiving end, the Endianness conversion is taken in charge. An important role of the COM component is to pack/unpack "signals" into/from "I-PDUs." Note that, as the maximal length of an I-PDU depends on the underlying networks, the design of a COM component has to take into account the networks and therefore it is not fully independent of the hardware architecture. The COM component has also to determine the points in time where to send the I-PDUs. This is based on the attributes Transmission Mode of an I-PDU and on the attribute Transfer Property of each signal that it contains. Table 13.1 summarizes the combinations that are possible. Note that the none Transmission Mode is not indicated in this table, in that case the transmission is driven by the underlying network layers.

TABLE 13.1 Transmission Mode of an I-PDU versus Transfer Property of Its Signals

Transfer Property of the signals Transmission mode of the I-PDU	All the signals in the I-PDU are triggered	All the signals in the I-PDU are pending	At lease one signal is triggered and one is pending in the I-PDU
Direct	The transmission of the I-PDU is done each time a signal is sent		This configuration could be dangerous: if no emission of triggered signals occurs, the pending signals will never be transmitted
Periodic	The transmission of the I-PDU is done periodically	The transmission of the I-PDU is done periodically	The transmission of the I-PDU is done periodically
Mixed	The transmission of the I-PDU is done each time a signal is sent and at each period	The transmission of the I-PDU is done periodically	The transmission of the I-PDU is done each time a Triggered signal is sent and at each period

The COM component is generated off-line on the basis of the knowledge of the signals, the I-PDUs, and the allocation of application software components on the ECUs. The AUTOSAR PDU Router (see Figure 13.9), according to the configuration, dispatches each I-PDU to the right network communication stack. This basic component is statically generated off-line as soon as the allocation of software components and the operational architecture is known. Other basic software components of the communication stack are responsible for the segmenting/reassembling of I-PDU(s) when needed (FlexRay TP, CAN TP) or for providing an interface to the communication drivers (FlexRay Interface, CAN Interface, LIN Interface).

13.4 Open Issues for Automotive Communication Systems

13.4.1 Optimized Networking Architectures

The traditional partitioning of the automotive application into several distinct functional domains with their own characteristics and requirements is useful in mastering the complexity, but this leads to the development of several independent subsystems with their specific architectures, networks, and software technologies.

Some difficulties arise from this partitioning as more and more cross-domain data exchanges are needed. This requires implementing gateways whose performances in terms of CPU load and impact on data freshness have to be carefully assessed (see, for instance, Ref. [67]). For instance, an ECU belonging, from a functional point of view, to a particular domain can be connected, for wiring reasons, onto a network of another domain. For example, the diesel particulate filter (DPF) is connected onto the body network in some vehicles even though it belongs, from a functional standpoint, to the powertrain. This can raise performance problems as the DPF needs a stream of data with strong temporal constraints coming from the engine controller located on the powertrain network. Numerous other examples of cross-domain data exchanges can be cited such as the engine controller (powertrain) that takes input from the climate control (body) or information from the powertrain displayed on the dashboard (body). There are also some functions that one can consider as being cross-domains such as the immobilizer, which belongs both to the body and powertrain domains. Upcoming X-by-Wire functions will also need very tight cooperation between the ECUs of the chassis, the powertrain, and the body.

A current practice is to transfer data between different domains through a gateway usually called the "central body electronic," belonging to the body domain. This subsystem is recognized as being critical in the vehicle: it constitutes a single point of failure, its design is overly complex and performance problems arise due to an increasing workload.

An initial foreseeable domain of improvement is to further develop the technologies needed for the interoperability between applications located on different subnetworks. With the AUTOSAR project, significant progresses in the area of MW have been achieved over the last years and we are coming closer to the desirable characteristics listed in Section 13.3.

Future work should also be devoted to optimizing networking architectures. This implies rethinking the current practice that consists of implementing networks on a per-domain basis. The use of technologies that could fulfill several communication requirements (e.g., high-speed, event-triggered and TT communication, all possible with FlexRay) with scalable performances is certainly one possible direction for facilitating the design. Certainly, software tools, such as our tool NETCAR-Analyzer (see http://www.realtimeatwork.com), will be helpful to master the complexity and come up with cost and dependability-optimized solutions. The use of software along the development cycle will be facilitated by the advent of the ASAM FIBEX standard [2], in the process of being adopted by AUTOSAR, which enables to fully describe the networks embedded in a vehicle (CAN, LIN, FlexRay, MOST, and TTCAN protocols), the frames that are exchanged between ECUs and the gatewaying strategies.

13.4.2 System Engineering

The verification of the performances of a communication system is twofold. On the one hand, some properties of the communication system services can be proved independently of the application. For instance, the correctness of the synchronization and the membership and clique avoidance services of TTP/C have been studied using formal methods in Refs. [5,6,50].

There are other constraints whose fulfillment cannot be determined without a precise model of the system. This is typically the case for real-time constraints on tasks and signals, where the patterns of activations and transmissions have to be identified. Much work has already been done in this field during the last 10 years: schedulability analysis on priority buses [68], joint schedulability analysis of tasks and messages [24,69], probabilistic assessment of the reliability of communications under EMI [19,20,42], etc. What is now needed is to extend these analyses to take into account the peculiarities of the platforms in use (e.g., overheads due to the OS and the stack of communication layers) and to integrate them in the development process of the system. The problem is complicated by the development process being shared between several partners (the carmaker and various third-party suppliers). Ways have to be found to facilitate the integration of components developed independently and to ensure their interoperability.

In terms of the criticality of the involved functions, future automotive X-by-Wire systems can reasonably be compared with Flight-by-Wire systems in the avionic field. According to Ref. [73], the probability of encountering a critical safety failure in vehicles must not exceed 5×10^{-10} per hour and per system, but other studies consider 10^{-9}. It will be a real challenge to reach such dependability, in particular, because of the cost constraints. It is certain that the know-how gathered over the years in the avionic industry can be of great help but design methodologies adapted to the automotive constraints have to be developed.

The first step is to develop technologies able to integrate different subsystems inside a domain (see Section 13.4.1), but a real challenge is to shift the development process from subsystem integration to a complete integrated design process. The increasing amount of networked control functions inside in-car embedded systems leads to developing specific design processes based, among others, on formal analysis and verification techniques of both dependability properties of the networks and dependability requirements of the embedded application.

References

1. A. Albert. Comparison of event-triggered and time-triggered concepts with regards to distributed control systems. In *Proceedings of Embedded World 2004*, Nürnberg, Germany, February 2004.
2. ASAM. FIBEX—field bus exchange format, version 3.0. Available at http://http://www.asam.net/, January 2008.
3. A. Avizienis, J. Laprie, and B. Randell. Fundamental concepts of dependability. In *Proceedings of the 3rd Information Survivability Workshop*, pp. 7–12, Boston, MA, 2000.
4. M. Ayoubi, T. Demmeler, H. Leffler, and P. Köhn. X-by-Wire functionality, performance and infrastructure. In *Proceedings of Convergence 2004*, Detroit, MI, 2004.
5. R. Barbosa and J. Karlsson. Formal specification and verification of a protocol for consistent diagnosis in real-time embedded systems. In *La Grande Motte, at the Third IEEE International Symposium on Industrial Embedded Systems (SIES'2008)*, France, June 2008.
6. G. Bauer and M. Paulitsch. An investigation of membership and clique avoidance in TTP/C. In *Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems*, Nuremberg, Germany, 2000.
7. P. Bühring. Safe-by-Wire Plus: Bus communication for the occupant safety system. In *Proceedings of Convergence 2004*, Detroit, MI, 2004.
8. L. Casparsson, A. Rajnak, K. Tindell, and P. Malmberg. Volcano—A revolution in on-board communications. Technical Report, Volvo, 1999.

9. G. Cena and A. Valenzano. Performance analysis of Byteflight networks. In *Proceedings of the 2004 IEEE Workshop of Factory Communication Systems (WFCS 2004)*, pp. 157–166, Vienna, Austria, September 2004.
10. FlexRay Consortium. FlexRay communications system—protocol specification—version 2.1. Available at http://www.flexray.com, December 2005.
11. Intel Corporation. Introduction to in-vehicle networking. Available at http://support.intel.com/design/ auto/autolxbk.htm, 2004.
12. R.I. Davis, A. Burns, R.J. Bril, and J.J. Lukkien. Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):239–272, April 2007.
13. P.H. Dezaux. Migration strategy of in-house automotive real-time applicative software in AUTOSAR standard. In *Proceedings of the 4th European Congress Embedded Real Time Software* (*ERTS 2008*), Toulouse, France, 2008.
14. P. Feiler. Real-time application development with OSEK—a review of OSEK standards, 2003. Technical Note CMU/SEI 2003-TN-004.
15. H. Fennel, S. Bunzel, H. Heinecke, J. Bielefeld, S. Fürstand, K.P. Schnelle, W. Grote, N. Maldenerand, T. Weber, F. Wohlgemuth, J. Ruh, L. Lundh, T. Sandén, P. Heitkämper, R. Rimkus, J. Leflour, A. Gilberg, U. Virnich, S. Voget, K. Nishikawa, K. Kajio, K. Lange, T. Scharnhorst, and B. Kunkel. Achievements and exploitation of the AUTOSAR development partnership. In *Convergence 2006*, Detroit, MI, October 2006.
16. J. Ferreira, P. Pedreiras, L. Almeida, and J.A. Fonseca. The FTT-CAN protocol for flexibility in safety-critical systems. *IEEE Micro, Special Issue on Critical Embedded Automotive Networks*, 22(4):46–55, July–August 2002.
17. Ford Motor Company. Ford to study in-vehicle electronic devices with advanced simulators. Available at url http://media.ford.com/article_display.cfm?article_id=7010, 2001.
18. S. Fürst. AUTOSAR for safety-related systems: Objectives, approach and status. In *2nd IEE Conference on Automotive Electronics*, London, U.K., March 2006. IEE.
19. B. Gaujal and N. Navet. Fault confinement mechanisms on CAN: Analysis and improvements. *IEEE Transactions on Vehicular Technology*, 54(3):1103–1113, May 2005.
20. B. Gaujal and N. Navet. Maximizing the robustness of TDMA networks with applications to TTP/C. *Real-Time Systems*, 31(1-3):5–31, December 2005.
21. M. Grenier, L. Havet, and N. Navet. Configuring the communication on FlexRay: The case of the static segment. In *ERTS Embedded Real Time Software 2008*, 2008.
22. M. Grenier, L. Havet, and N. Navet. Pushing the limits of CAN—Scheduling frames with offsets provides a major performance boost. In *ERTS Embedded Real Time Software 2008*, 2008.
23. H. Kopetz et al. *Specification of the TTP/A Protocol*. University of Technology Vienna, Vienna, Austria, September 2002.
24. R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst. System level performance analysis—The SymTA/S approach. *IEE Proceedings Computers and Digital Techniques*, 152(2):148–166, March 2005.
25. International Standard Organization. *ISO 11519-2, Road Vehicles—Low Speed Serial Data Communication—Part 2: Low Speed Controller Area Network*. ISO, 1994.
26. International Standard Organization. *ISO 11519-3, Road Vehicles—Low Speed Serial Data Communication—Part 3: Vehicle Area Network (VAN)*. ISO, 1994.
27. International Standard Organization. *ISO 11898, Road Vehicles—Interchange of Digital Information—Controller Area Network for High-Speed Communication*. ISO, 1994.
28. International Standard Organization. *15765-2, Road Vehicles—Diagnostics on CAN—Part 2: Network Layer Services*. ISO, 1999.
29. International Standard Organization. *11898-4, Road Vehicles—Controller Area Network* (*CAN*)*—Part 4: Time-Triggered Communication*. ISO, 2000.

30. K. Johansson, M. Törngren, and L. Nielsen. Vehicle applications of controller area network. In D. Hristu-Varsakelis and W.S. Levine, eds., *Handbook of Networked and Embedded Control Systems*. Birkhäuser, Boston, MA, 2005.
31. P. Koopman. Critical embedded automotive networks. *IEEE Micro, Special Issue on Critical Embedded Automotive Networks*, 22(4):14–18, July–August 2002.
32. M. Krug and A.V. Schedl. New demands for in-vehicle networks. In *Proceedings of the 23rd EUROMICRO Conference'97*, Budapest, Hungary, July 1997.
33. G. Leen and D. Heffernan. Expanding automotive electronic systems. *IEEE Computer*, 35(1): 88–93, January 2002.
34. LIN Consortium. *LIN Specification Package, Version 1.3*, December 2002. Available at http://www.lin-subbus.org/.
35. LIN Consortium. *LIN Specification Package, Revision 2.1*, November 2006. Available at http://www.lin-subbus.org/.
36. Y. Martin. L'avenir de l'automobile tient à un fil. *L'argus de l'automobile*, 3969:22–23, March 2005.
37. E. Mayer. Serial bus systems in the automobile—part 5: MOST for transmission of multimedia data. Summary of Networking Competence, February 2008. Published by Vector Informatik GmbH.
38. B. Müller, T. Führer, F. Hartwich, R. Hugel, and H. Weiler. Fault tolerant TTCAN networks. In *Proceedings of the 8th International CAN Conference (iCC)*, Las Vegas, NV, 2002.
39. MOST Cooperation. *MOST Specification, Revision 2.3*, August 2004. Available at http://www.mostnet.de.
40. H. Muyshondt. Consumer and automotive electronics converge: Part 1—Ethernet, USB, and MOST. Available at http://www.automotivedesignline.com/, February 2007.
41. H. Muyshondt. Consumer and automotive electronics converge: Part 2—a MOST implementation. Available at http://www.automotivedesignline.com/, March 2007.
42. N. Navet, Y. Song, and F. Simonot. Worst-case deadline failure probability in real-time applications distributed over CAN (Controller Area Network). *Journal of Systems Architecture*, 46(7):607–617, 2000.
43. N. Navet and Y.-Q. Song. Validation of real-time in-vehicle applications. *Computers in Industry*, 46(2):107–122, November 2001.
44. OSEK Consortium. *OSEKtime OS, Version 1.0*, July 2001. Available at http://www.osek-vdx.org/.
45. OSEK Consortium. *OSEK/VDX Fault-Tolerant Communication, Version 1.0*, July 2001. Available at http://www.osek-vdx.org/.
46. OSEK Consortium. *OSEK/VDX Communication, Version 3.0.3*, July 2004. Available at http://www.osek-vdx.org/.
47. OSEK Consortium. *OSEK/VDX Operating System, Version 2.2.2*, July 2004. Available at http://www.osek-vdx.org/.
48. M. Peteratzinger, F. Steiner, and R. Schuermans. Use of XCP on FlexRay at BMW. Translated reprint from HANSER Automotive 9/2006. Available at url https://www.vector-worldwide.com/vi_downloadcenter_en,,223.html?product=xcp, 2006.
49. H. Pfeifer. Formal methods in the automotive domain: The case of TTA. In N. Navet and F. Simonot-Lion, eds., *Automotive Embedded Systems Handbook*. CRC Press/Taylor and Francis, Boca Raton, FL, December 2008.
50. H. Pfeifer and F.W. von Henke. Formal analysis for dependability properties: The time-triggered architecture example. In *Proceedings of the 8th IEEE International Conference on Emerging Technologies and Factory Automation* (*ETFA 2001*), pp. 343–352, Antibes-Juan les Pins, France, October 2001.
51. J. Pimentel, J. Proenza, L. Almeida, G. Rodriguez-Navas, M. Barranco, and J. Ferreira. Dependable automotive CAN networks. In N. Navet and F. Simonot-Lion, eds., *Automotive Embedded Systems Handbook*. CRC Press/Taylor and Francis, Boca Raton, FL, December 2008.
52. L.M. Pinho and F. Vasques. Reliable real-time communication in can networks. *IEEE Transactions on Computers*, 52(12):1594–1607, 2003.

53. S. Poledna, W. Ettlmayr, and M. Novak. Communication bus for automotive applications. In *Proceedings of the 27th European Solid-State Circuits Conference*, Villach, Austria, September 2001.
54. T. Pop, P. Pop, P. Eles, and Z. Peng. Bus access optimisation for FlexRay-based distributed embedded systems. In *Proceedings of the conference on Design, Automation and Test in Europe (DATE '07)*, pp. 51–56, San Jose, CA, 2007. EDA Consortium.
55. A. Rajnák. Volcano—Enabling correctness by design. In R. Zurawski, ed., *The Embedded Systems Handbook*. CRC Press, Boca Raton, FL, August 2005.
56. A. Rajnák. The LIN Standard. In R. Zurawski, ed., *The Industrial Communication Technology Handbook*. CRC Press, Boca Raton, FL, January 2005.
57. A. Rajnák and M. Ramnefors. The Volcano communication concept. In *Proceedings of Convergence 2002*, Detroit, MI, 2002.
58. K. Ramaswamy and J. Cooper. Delivering multimedia content to automobiles using wireless networks. In *Proceedings of Convergence 2004*, Detroit, MI, 2004.
59. Robert Bosch GmbH. Time triggered communication on CAN: TTCAN. Available at http://www.semiconductors.bosch.de/en/20/ttcan/index.asp, 2008.
60. J. Rushby. A comparison of bus architecture for safety-critical embedded systems. Technical report, NASA/CR, March 2003.
61. R. Saket and N. Navet. Frame packing algorithms for automotive applications. *Journal of Embedded Computing*, 2:93–102, 2006.
62. A. Schedl. Goals and Architecture of FlexRay at BMW. Slides presented at the Vector FlexRay Symposium, March 2007.
63. B. Schätz, C. Kühnel, and Gonschorek. The FlexRay protocol. In N. Navet and F. Simonot-Lion, eds., *Automotive Embedded Systems Handbook*. CRC Press/Taylor and Francis, Boca Raton, FL, December 2008. ISBN 9780849380266.
64. Society of Automotive Engineers. J2056/1 class C application requirements classifications. In *SAE Handbook*. 1994.
65. Society of Automotive Engineers. J2056/2 survey of known protocols. In *SAE Handbook*, Vol. 2. 1994.
66. Society of Automotive Engineers. Class B data communications network interface—SAE J1850 standard—Rev. November 96, 1996.
67. J. Sommer and R. Blind. Optimized resource dimensioning in an embedded CAN–CAN gateway. In *IEEE Second International Symposium on Industrial Embedded Systems (SIES'2007)*, pp. 55–62, Lisbon, Portugal, July 2007.
68. K. Tindell, A. Burns, and A.J. Wellings. Calculating controller area network (CAN) message response times. *Control Engineering Practice*, 3(8):1163–1169, 1995.
69. K. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessors and Microprogramming*, 40:117–134, 1994.
70. TTTech Computertechnik GmbH. *Time-Triggered Protocol TTP/C, High-Level Specification Document, Protocol Version 1.1*, November 2003. Available at http://www.tttech.com.
71. M. Waern. Evaluation of protocols for automotive systems. Master's thesis, KTH Machine Design, Stockholm, Sweden, 2003.
72. C. Wilwert, N. Navet, Y.-Q. Song, and F. Simonot-Lion. Design of automotive X-by-Wire systems. In R. Zurawski, ed., *The Industrial Communication Technology Handbook*. CRC Press, Boca Raton, FL, January 2005.
73. X-by-Wire Project, Brite-EuRam 111 Program. X-By-Wire—Safety related fault tolerant systems in vehicles, Final Report, 1998.

14

Time-Triggered Communication

14.1 Time- and Event-Triggered Communication 14-1
14.2 Time-Triggered Communication in the Automotive Domain 14-2
14.3 Dependability Concepts 14-2
Fault, Error, and Failure • Fault Containment
14.4 Fundamental Services of a Time-Triggered Communication Protocol 14-3
Clock Synchronization • Periodic Exchange of State Messages • Fault Isolation Mechanisms • Diagnostic Services
14.5 Time-Triggered Communication Protocols 14-6
TTP/C • TTP/A • TTCAN • TT Ethernet
References 14-13

Roman Obermaisser
Vienna University of Technology

14.1 Time- and Event-Triggered Communication

It has been recognized that communication protocols fall into two general categories with corresponding strengths and deficiencies: event- and time-triggered control. Event-triggered protocols (e.g., TCP/IP, CAN, Ethernet, ARINC629) offer flexibility and resource efficiency. Time-triggered protocols (e.g., TTP/C, FlexRay) excel with respect to predictability, composability, error detection, and error containment.

In event-triggered architectures, the system activities, such as sending a message or starting computational activities, are triggered by the occurrence of events in the environment or the computer system. In time-triggered architectures (TTA), activities are triggered by the progression of global time. The major contrast between event- and time-triggered approaches lies in the location of control. Time-triggered systems exhibit autonomous control and interact with the environment according to an internal predefined schedule, whereas event-triggered systems are under the control of the environment and must respond to stimuli as they occur.

The time-triggered approach is generally preferred for safety-critical systems [Kop95,Rus01]. For example, in the automotive industry a TTA will provide the ability to handle the communication needs of by-wire cars [Bre01]. In addition to hard real-time performance, TTAs help in managing the complexity of fault-tolerance and corresponding formal dependability models, as required for the establishment of ultrahigh reliability (failure rates in the order of 10^{-9} failures/h). The predetermined points in time of the periodic message transmissions allow error detection and establishing of membership information. Redundancy can be established transparently to applications, i.e., without any modification of the function and timing of application systems. A time-triggered system also supports replica determinism [Pol96], which is essential for establishing fault-tolerance through active redundancy.

Furthermore, time-triggered systems support temporal composability [KO02] via a precise specification of the interfaces between subsystems, both in the value domain and temporal domain. The communication controller in a time-triggered system decides autonomously when a message is transmitted. The communication network interface (CNI) is a temporal firewall which isolates the temporal behavior of the host and the rest of the system.

14.2 Time-Triggered Communication in the Automotive Domain

With technologies such as AUTOSAR [AUT06], the automotive domain is moving from the federated architectural paradigm to the integrated architectural paradigm. Among the primary goals of AUTOSAR are the standardization of the basic software of an ECU (called standard core) and the ability to integrate software components from multiple suppliers. When integrating multiple software components in a shared-distributed computer system, emphasis must be placed on avoiding emerging complexity through unintended interference of software components and application subsystems on the shared platform [Kop08].

For a deeper understanding, consider an exemplary scenario with two subsystems. If the two subsystems share a common CAN bus [Bos91], then both subsystems must be analyzed and understood to reason about the correct behavior of any of the two subsystems. As the message transmissions of one subsystem can delay message transmission of the other application subsystems, arguments concerning the correct temporal behavior must be based on an analysis of both subsystems. In a totally federated system, on the other hand, unintended side effects are ruled out, because the two subsystems are assigned to separate computer system.

Time-triggered communication protocols support the management of the complexity in an integrated architecture such as AUTOSAR as follows:

- *Rigorous encapsulation and error containment*: Based on static resource allocations at the communication network (i.e., a time division multiple access [TDMA] scheme) and the operating system (i.e., fixed cyclic scheduler), time-triggered communication protocols ensure that a software component cannot affect the communication resources that are available to other software components.
- *Predictability and determinism*: The rigorous encapsulation simplifies a timing analysis of application subsystems and software components. When determining the temporal behavior of a software component, a designer can abstract from the behavior of other software components, because the time-triggered communication protocol guarantees to each software component predefined communication resources.
- *Foundation for realization of safety-critical applications*: Time-triggered communication protocols offer services (e.g., fault-tolerance, diagnostic information) that are the foundation for the achievement of sufficient dependability for safety-critical applications (e.g., X-by-wire). The realization of these services is challenging and can become a primary sources of faults and unreliability when handled in an ad-hoc manner [Mac88].

14.3 Dependability Concepts

Dependability is the ability of a computing system to deliver services that can justifiably be trusted [Car82]. In the following, key concepts of dependability will be described that are used in the reminder of this chapter.

14.3.1 Fault, Error, and Failure

A failure [Lap92] occurs when the delivered service deviates from fulfilling the functional specification. An error is that part of the system state which is liable to lead to a subsequent failure. A failure occurs when the error reaches the service interface. A fault is the adjudged or hypothesized cause of an error. As stated in Ref. [ALR00], the concept of fault is introduced to stop recursion.

Due to the recursive definition of systems, a failure at a particular level of decomposition can be interpreted as a fault at the next upper level of decomposition, thereby leading to a hierarchical causal chain.

14.3.2 Fault Containment

A fault containment region (FCR) is defined as a subsystem that operates correctly regardless of any arbitrary logical or electrical fault outside the region [LH94]. The justification for building ultra-reliable systems from replicated resources rests on an assumption of failure independence among redundant units. For this reason, the independence of FCRs is of critical importance [BCV91]. The independence of FCRs can be compromised by shared physical resources (e.g., power supply, timing source), external faults (e.g., electromagnetic interference, spatial proximity), and design.

14.3.2.1 Error Containment

Although an FCR can restrict the immediate impact of a fault, fault effects manifested as erroneous data can propagate across FCR boundaries using the communication system. For this reason, the system must also provide error containment [LH94] to avoid error propagation through message failures. A message failure can be either a message value failure or a message timing failure [CASD85]. A message value failure occurs if the data contained in a message are incorrect. A message timing failure means that the message send or receive instants are not in agreement with the specification.

Error containment involves an independent component for error detection and mediation of a component's access to the shared network. The error detection and mediation mechanisms must be part of a different FCR than the message sender [Kop03]. Otherwise, the error containment mechanisms may be impacted by the same fault that caused the message failure.

One can distinguish two types of error containment [Rus99]:

- *Spatial Partitioning*: Spatial partitioning ensures that software in one FCR cannot alter the code or private data of another FCR. Spatial partitioning also prevents an FCR from interfering with control of external devices (e.g., actuators) of other FCRs.
- *Temporal Partitioning*: Temporal partitioning ensures that an FCR cannot affect the ability of other FCRs to access shared resources, such as the common network or a shared CPU. This includes the temporal behavior of the services provided by resources (latency, jitter, duration of availability during a scheduled access).

The probability for preventing the propagation of the consequences of an error is called the error-containment coverage. Error containment is a prerequisite for building fault-tolerant systems as without error containment a single fault has the ability to corrupt the whole system.

14.4 Fundamental Services of a Time-Triggered Communication Protocol

In the following, four fundamental services of a time-triggered communication protocol are explained, namely, clock synchronization, the periodic exchange of state messages, fault isolation, and diagnostic services.

14.4.1 Clock Synchronization

Due to clock drifts, the clock times in an ensemble of clocks will drift apart, if clocks are not periodically resynchronized. Clock synchronization is concerned with bringing the values of clocks in close relation with respect to each other. A measure for the quality of synchronization is the precision. An ensemble of clocks that are synchronized to each other with a specified precision offers a global time. The global time of the ensemble is represented by the local time of each clock, which serves as an approximation of the global time [KO87].

Two important parameters of a global time base are the granularity and horizon. The granularity determines the minimum interval between two adjacent ticks of the global time (also called macrogranule), i.e., the smallest interval that can be measured with the global time. The horizon determines the instant when the time will wrap around.

The reasonableness condition [Kop97] for a global time base ensures that the synchronization error is bounded to less than one macrogranule. However, due to the synchronization and digitalization error, it is impossible to establish the temporal order of occurrences based on their timestamp, if timestamps differ by only a single tick. A solution to this problem is the introduction of a sparse time base [Kop92].

14.4.2 Periodic Exchange of State Messages

A time-triggered communication system is designed for the periodic exchange of messages carrying state information. These messages are called "state messages."

Information with state semantics contains the absolute value of a real-time entity (e.g., temperature in the environment is 41 °C). The self-contained nature and idempotence of state messages ease the establishment of state synchronization, which does not depend on exactly-once processing guarantees. As applications are often only interested in the most recent value of a real-time object, old state values can be overwritten with newer state values. Hence, a time-triggered communication system does not require message queues.

The periodic transmission of state messages is triggered by the progression of the global time according to a TDMA scheme. TDMA statically divides the channel capacity into a number of slots and assigns a unique slot to every node. The communication activities of every node are controlled by a time-triggered communication schedule. The schedule specifies the temporal pattern of messages transmissions, i.e., at what points in time nodes send and receive messages. A sequence of sending slots, which allows every node in an ensemble of n nodes to send exactly once, is called a TDMA round. The sequence of the different TDMA rounds forms the cluster cycle and determines the periodicity of the time-triggered communication.

The a priori knowledge about the times of message exchanges enables the communication system to operate autonomously. The temporal control of communication activities is within the sphere of control of the communication system. Hence, the correct temporal behavior of the communication system is independent of temporal behavior of the application software in the host computer and can be established in isolation.

14.4.3 Fault Isolation Mechanisms

A time-triggered communication protocol and the corresponding system architecture need to provide rules for partitioning a system into independent FCRs. In a system with active redundancy, the dependencies among FCRs (i.e., correlation between failure probability of FCRs) have a significant impact on the system reliability [OKS07]. The independence of FCRs can be compromised by shared physical resources (e.g., hardware, power supply, time base), by external faults (e.g., electromagnetic interference, heat, shock, spatial proximity), and by design faults.

In a time-triggered communication protocol, the error containment mechanisms for timing message failures can be enforced transparently to the application. Using the a priori knowledge concerning the global points in time of all intended message sent and received instants, autonomous guardians can block timing message failures. For this purpose, node-local and centralized guardians have been developed and validated for different time-triggered communication protocols (e.g., [BKS03,Gua05]). For example, in TTP, a guardian transforms a message which it judges untimely into a syntactically incorrect message by cutting off its tail [Kop03].

Error containment for value message failures is generally not part of a time-triggered communication protocol, but within the responsibility of the host computers. For example, value failure detection and correction can be performed using N-modular redundancy (NMR). N replicas receive the same requests and provide the same service. The output of all replicas is provided to a voting mechanism, which selects one of the results (e.g., based on majority) or transforms the results to a single one (average voter). The most frequently used N-modular configuration is triple-modular redundancy (TMR). By employing three components and a voter, a single consistent value failure in one of the constituting components can be tolerated.

Although not natively provided by time-triggered communication protocols, NMR is enabled by time-triggered communication protocols by supporting replica determinism [Pol96]. Fault-free replicated components exhibit replica determinism, if they deliver identical outputs in an identical order within a specified time interval. Replica determinism simplifies the implementation of fault-tolerance by active redundancy, as failures of components can be detected by carrying out a bit-by-bit comparison of the results of replicas. Replica nondeterminism is introduced either by the interface to the real world or the system's internal behavior.

14.4.4 Diagnostic Services

Diagnostic services are concerned with the identification of failed subsystems. Diagnostic services can trigger the autonomous recovery of a system in case of a transient subsystem failure. In addition, diagnostic services can support the replacement of defective subsystems if a failure is permanent.

An example of a diagnostic services that can be found in time-triggered communication protocols is a solution to the membership problem. The membership problem is a fundamental problem in distributed computing, because it allows solutions to other important problems in designing fault-tolerant systems [GP96]. The membership problem is defined as the problem of achieving agreement on the identity of all correctly functioning processes of a process group. A process is correct, if its behavior complies with the specification. Otherwise the process is denoted as faulty.

In the context of integrated architecture, it makes sense to establish membership information for FCRs, as FCRs can be expected to fail independently. Depending on the assumed types of faults, an FCR is either an entire system component or a subsystem within a component (e.g., a task) dedicated to a function.

A service that implements an algorithm for solving the membership problem and offers consistent membership information is called a membership service. A membership service simplifies the provision of many application algorithms, as the architecture offers generic error detection capabilities via this service. Applications can rely on the consistency of the membership information and react to detected failures of FCRs as indicated by the membership service.

A membership service also plays an important role for controlling application level fault-tolerance mechanisms that deal with failures of functions. If a function fails—as more FCRs have failed than can be tolerated by the given amount of redundancy—all that an integrated architecture can do is to inform other functions about this condition so they can react accordingly by application level fault-tolerance mechanisms.

In a time-triggered communication system, the periodic message send times are membership points of the sender [KGR91]. Every receiver knows a priori when a message of a sender is supposed

to arrive, and interprets the arrival of the message as a life sign at the membership point of the sender. From the arrival of the expected messages at two consecutive membership points, it can be concluded that the sender was operational during the interval delimited by these membership points.

14.5 Time-Triggered Communication Protocols

This section gives an overview of four important representatives of time-triggered communication protocols: TTP/C, TTP/A, TTCAN, and TT Ethernet.

14.5.1 TTP/C

TTP/C is part of the TTA, a system architecture for the design and implementation of dependable distributed real-time systems [BK00]. TTP/C provides four services: a predictable fault-tolerant time-triggered transport service, clock synchronization, error containment, and a membership service.

The basic building block of the TTA is a node computer, which is a self-contained composite hardware/software subsystem (system component). A cluster is a set of nodes that are interconnected by two redundant communication channels. For the interconnection of nodes, the TTA distinguishes between two physical interconnection topologies, namely, a TTA-bus and a TTA-star (see Figure 14.1).

A TTA-bus consists of replicated passive busses. Every node is connected to two independent guardians, which use the a priori knowledge about the points in time of communication activities to prevent communication outside a node's slot. In the TTA-star topology, the interconnection of nodes occurs via two replicated central guardians. The star topology has the advantage of a higher level of independence, as guardians are located at a physical distance from nodes. Furthermore, guardians reshape signals and support additional monitoring services [ASBT03].

The TTP/C protocol uses TDMA to control the media access to the two independent communication channels. Time is partitioned into slots, which are statically assigned to nodes. A sequence of slots that allows each node to send a message forms a TDMA round. A sequence of predefined TDMA rounds is called a cluster cycle. The cluster cycle also defines the periodicity of the message transmissions. Information about the points in time of all message transmissions during a cluster cycle are contained in the message schedule.

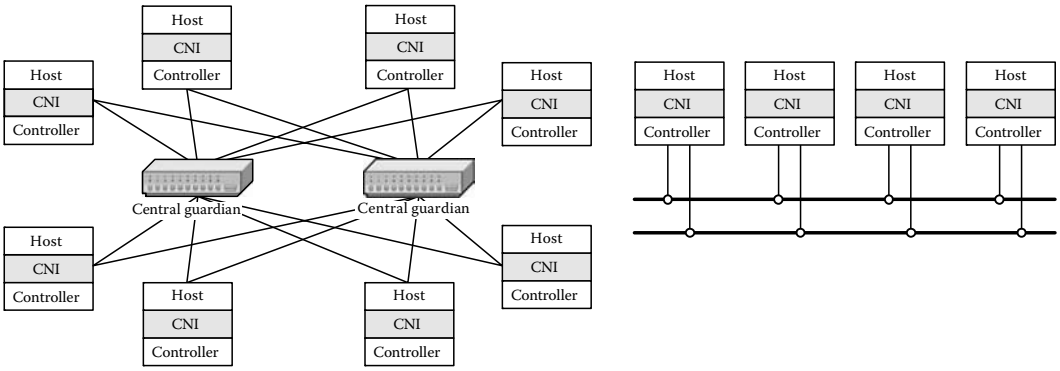


FIGURE 14.1 TTP/C network with star topology (left) and bus topology (right).

14.5.1.1 Clock Synchronization

TTP/C provides fault-tolerant clock synchronization via the Fault-Tolerant Average clock synchronization algorithm [LL84]. The clock synchronization algorithm of the TTA has been formally verified in Ref. [PSF99]. It differs from other algorithms by the fact that no special synchronization messages are used for the exchange of the local clock values of nodes. The difference between the expected and the actual arrival time of an incoming message is used to estimate the deviation between the local clock of the receiver and the sender. Furthermore, TTP/C provides support to collect timing information only from selected nodes, thereby nodes with inferior oscillators can be left out as inputs for clock synchronization.

14.5.1.2 Periodic Exchange of State Messages

TTP/C distinguishes between two types of messages, namely, initialization frames (i-frames) and normal frames (n-frames). i-Frames carry the part of the controller state, which is required for the startup and the reintegration of nodes. This subset of the controller state is denoted as c-state and includes the current position in the communication schedule, the global time, and the membership vector. N-frames are used during normal operation and carry application data. Both i- and n-frames are protected by cyclic redundancy code (CRC) checks. In n-frames, CRC checking is also used for enforcing agreement on the controller states. The sender calculates the CRC of an n-frame over the message contents and the c-state of the sender. At the receiver, the CRC of an n-frame is calculated over the received message contents and the c-state of the receiver. Consequently, different c-states at the receiver and the sender will produce a negative result of the CRC check and result in a message omission failure.

14.5.1.3 Fault Isolation Mechanisms

Fault containment in TTP/C is achieved by proper architectural design decisions concerning resource sharing to limit the impact of a single fault to a single FCR [Kop03]. The fault hypothesis of TTP/C [Kop04] regards a complete node computer as an FCR because of the shared hardware resources. The shared resources of a node computer include the computing hardware, the power supply, the timing source, or the physical space. Consequently, there is a nonnegligible probability that a fault in any one of these resources will affect the entire node computers.

Based on the fault containment, TTP/C prevents timing-error propagation out of an FCR through guardians (e.g., using an intelligent star coupler [KBP01]). Error containment for value-failure is performed at the application level on top of the TTP/C communication protocol, e.g., using TMR.

14.5.1.4 Diagnostic Services

In addition, TTP/C includes a membership service. The membership service provides nodes with consistent information about the operational state of every node in the cluster. In case multiple cliques with different membership views form, clique avoidance ensures that the minority cliques leave the membership [BP00].

14.5.1.5 Commercial or Prototypical Components

TTP/C-based products are available from the company TTTech Computertechnik AG http://www.tttech.com/. For example, the "TTP Powernode" [TTT02a] is equipped with a Motorola embedded Power PC processor (MPC555 Black Oak) and the TTP-C2 controller AS8202NF. Through its on-chip FPU, this host CPU is suited for simulation purposes and code generated from Matlab/Simulink. The TTP Powernode is flexible and provides interfaces to a 2-channel

FIGURE 14.2 TTP monitoring node.

fault-tolerant TTP/C bus with 5 Mbps. In addition, the TTP Powernode provides interfaces to CAN and TTP/A. As an operating system, TTPOS [Tan99] with the fault-tolerance layer FT-COM stack can be used.

Another TTP/C-based product is the TTP monitoring node [TTT02c] (c.f. Figure 14.2), which is also based on the TTP-C2 controller. The TTP monitoring node is equipped with the Motorola embedded Power PC processor MPC855T, which features an on-chip fast Ethernet controller (100BASE-T), which can be used, e.g., for TCP/IP connections to a PC. The monitoring node supports synchronous (25 Mbps) and asynchronous (MFM—5 Mbps) bus interfaces for establishing the TTP/C communication. For the synchronous bus interface, it employs a 100BASE-TX physical layer with the media independent interface (MII) forming the bus toward the physical interfaces. Furthermore, a PCMCIA slot for cards type I/II/III is available on the monitoring nodes.

The MPC855T host CPU employs a PowerPC core and runs at 80 MHz. It is equipped with 16 MBytes of RAM, and 8 MBytes of flash memory. The CNI memory of the TTP-C2 controller is mapped into the address space of the MPC855T host CPU.

The TTP monitoring node uses the embedded real-time Linux variant RTAI as its operation system. RTAI [BBD+00] is short for "Real Time Application Interface". It is a real-time Linux extension developed by the Dipartimento di Ingegneria Aerospaziale Politecnico di Milano, which combines a real-time hardware abstraction layer (RTHAL) with a real-time application interface for making Linux suitable for real-time applications. RTAI introduces a real-time scheduler, which runs the conventional Linux operating system kernel as the idle task, i.e., non-real-time Linux only executes when there are no real-time tasks to run, and the real-time kernel is inactive.

14.5.2 TTP/A

TTP/A [EHK+05] is a member of the TTP protocol family that is intended for low-cost networks of sensors and actuators. TTP/A uses a structured name space as the source and sink of all communication activities. This structured name space is called interface file system (IFS). An IFS file is an indexed sequential file with up to 256 records. A record has a fixed length of 4 bytes (32 bits).

14.5.2.1 Clock Synchronization

In TTP/A, communication is organized into rounds each consisting of several TDMA slots. A slot is the unit for transmission of exactly one data byte transmitted via standard universal asynchronous receiver transmitter (UART) format. Each communication round is started by the master with a so-called fireworks byte. The fireworks byte serves as the epoch of the global time base and provides a reference signal for clock synchronization. The fireworks byte enables slave nodes to synchronize their local clocks. During startup, a fireworks byte with a regular bit pattern is used to establish initial synchronization in conjunction with slave nodes containing imprecise on-chip oscillators.

14.5.2.2 Periodic Exchange of State Messages

The TTP/A protocol distinguishes two types of communication rounds, namely "multipartner rounds" and "master/slave rounds." Multipartner rounds serve the purpose of periodically transmitting messages with state information. A multipartner round consists of a configurable number of slots with a specific sender node for each slot. The configuration of a round is defined at design time using a static data structure. This data structure defines which node transmits in a slot, the operation of each individual slot, and the receiving nodes. As the current global time in a TTP/A system unambiguously identifies a message, no explicit message name (e.g., message identifier) is required in messages.

Master–slave rounds allow to read data from an IFS file record, to write data to an IFS file record, or to execute a selected IFS file record within the cluster. A master–slave round consists of two phases, an address phase, and a data phase. During the address phase the master specifies (in a message to the slave node) which type of file operation is intended (read, write, or execute) and the address of the selected file record. Master–slave rounds can be employed for inherently event-triggered activities like monitoring, maintenance, or configuration. A maintenance engineer can open an event-triggered communication channel to a node by using corresponding tools as described in Ref. [Pit02].

14.5.2.3 Fault Isolation Mechanisms

TTP/A focuses on low-cost fieldbus networks, thus providing limited error detection and fault isolation. TTP/A includes a parity bit in every byte and offers the flexibility of byte-protected, nibble protected, or unprotected data fields. TTP/A assumes that the application will provide an application-specific end-to-end protection for data fields that are not protected by the protocol.

TTP/A does not employ bus guardians and assumes that faulty nodes will exhibit a single simple external failure mode: fail-silence. The resulting assumption coverage is adequate for non-safety-critical, low-cost fieldbus applications.

14.5.2.4 Diagnostic Services

TTP/A specifies a Diagnostic and Maintenance (DM) interface, which opens a communication channel to the internals of a node for the purpose of DM. This interface is used to set parameters and to retrieve information about the internals of a node, e.g., for the purpose of fault diagnosis. The maintenance engineer who accesses the sensor internals via the DM interface is required to have detailed knowledge about the internal structure and behavior of the node. The DM interface is available during system operation without disturbing the real-time service. To achieve the objective of supporting online maintenance without probe-effects, the sporadic diagnostic traffic is implemented using master–slave rounds. Thereby, it coexists with the real-time traffic (which is implemented using multipartner rounds) without disturbing the latter.

14.5.2.5 Commercial or Prototypical Components

Several prototype implementations are available for embedded computing nodes based on the TTP/A protocol. For example, Ref. [Elm06] describes an implementation of a TTP/A node using the 8-bit microcontroller Atmel AVR AT90S4433 (cf. Figure 14.3). This microcontroller has 4 KByte of Flash memory and 128 Bytes of RAM. The physical network interface is implemented using an ISO 9141 k-line bus, namely, a single wire bus with a bandwidth up to 50 kbps. In addition, implementations using other Atmel microcontrollers (e.g., Atmel AVR AT90S8515 or AT91X40) and Microchip PIC microcontrollers (e.g., PIC 16F874) have been realized [Tro02].

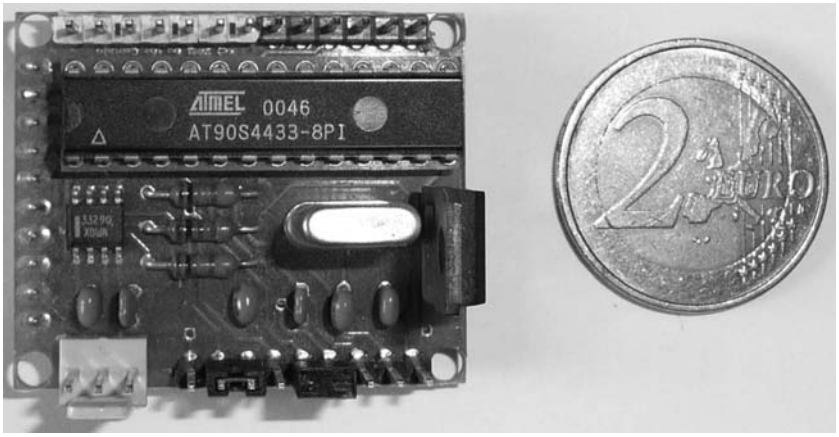


FIGURE 14.3 TTP/A node.

14.5.3 TTCAN

The Time-Triggered CAN (XCTL) [TBW+00] protocol is an example for time-triggered services, which are layered on top of an event-triggered protocol. XCTL is based on the CAN data link layer as specified in ISO 11898 [Int93].

14.5.3.1 Clock Synchronization

XCTL is a master–slave protocol that uses a time master for initiating communication rounds. XCTL employs multiple time masters for supporting fault-tolerance. The time master periodically sends a reference message, which is recognized by clients via its identifier. The period between two successive reference messages is called "basic cycle." The reference message contains information about the current time of the time master as well as the number of the cycle.

To improve the precision of the global time base, slave nodes measure the duration of the basic cycle and compare this measured duration with the values contained in the reference messages. The difference between these measured duration and the nominal duration (as indicated by the time master) is used for drift correction.

14.5.3.2 Periodic Exchange of State Messages

The basic cycle can consist of three types of windows, namely, exclusive time windows, free time windows, and arbitrating time windows.

1. *An exclusive time window* is dedicated to a single periodic message. A statically defined node sends the periodic message within an exclusive time window. An offline design tool ensures that no collisions occur.
2. *The arbitrating time windows* are dedicated to spontaneous messages. Multiple nodes can compete for transmitting a message within an arbitrating time window. The bitwise arbitration mechanism of CAN decides which message actually succeeds.
3. *Free time windows* are reserved for further extensions. When required, free time windows can be reconfigured as exclusive or arbitrating time windows.

Retransmissions are prohibited in all three types of windows.

14.5.3.3 Fault Isolation Mechanisms

To tolerate the failure of a single channel bus, channel redundancy has been proposed for TTCAN [MFH+02]. This solution uses gateway nodes, which are connected to multiple channel busses, to synchronize the global time of the different busses and maintain a consistent schedule. However, no analysis of the dependencies between the redundant busses has been formed. For example, due to the absence of bus guardians, a babbling idiot failure [Tem98] of a gateway node has the potential to disrupt communication across multiple redundant busses.

14.5.3.4 Diagnostic Services

The CAN protocol, which forms the basis for TTCAN, has built-in error detection mechanisms such as CRC codes to detect faults on the communication channel or error counters.

In addition, the a priori knowledge of the time-triggered send and receive instants can be used to establish a membership service. Although such a service is not mandatory in TTCAN, Ref. [BKAS] describes the design and implementation of a membership service on top of TTCAN.

14.5.3.5 Commercial or Prototypical Components

TTCAN controllers are available both as IP cores and as part of microcontrollers. For example, the TTCAN IP Module is available from Bosch [Bos06]. This IP module is described in very high speed integrated circuit hardware description language (VHDL) and can be integrated as a stand-alone device or as part of an application specific integrated circuit (ASIC). The IP module realizes a state machine for the ISO 11898-4 time-triggered communication. The state machine uses the reference messages of the CAN bus as a synchronizing event to establish a global time base and perform message transmissions according to the a priori known communication schedule.

Another IP module supporting TTCAN is the MultiCAN IP of Infineon Technologies [Kel03], which is deployed in several Infineon devices (e.g., TC1130 [Inf03]). Like the TTCAN IP Module, the MultiCAN IP performs clock synchronization and offers a scheduler to trigger the predefined message transmissions.

In addition, microcontrollers are available that support the implementation of TTCAN. For example, Atmel provides microcontrollers (e.g., T89C51CC01, T89C51CC02 [Atm08a,Atm08b]) that permit the implementation of TTCAN using higher software layers. In particular, these microcontrollers offer a single shot transmission mode to prevent automatic retransmission upon errors. This transmission mode is required for the implementation of TTCAN.

14.5.4 TT Ethernet

Time-triggered Ethernet [KAGS05] is a protocol that unifies real-time and non-real-time traffic into a single coherent communication architecture. Time-triggered Ethernet introduces time-triggered messages with temporal guarantees, while also supporting event-triggered standard Ethernet communication to ensure compatibility to standard Ethernet commercial of-the-shelf components.

14.5.4.1 Clock Synchronization

TT Ethernet is based on a uniform 64-bit binary time-format that is based on the physical second. Fractions of a second are represented as 24 negative powers of two (down to about 60 ns), and full seconds are presented in 40 positive powers of two (up to about 30,000 years). This time format has been standardized by the object management group (OMG) in the smart transducer interface standard [EHK+05]. The time format of TT Ethernet is closely related to the time-format of the General

Positioning System (GPS) time, but has a wider horizon. The similarity to the GPS time-format facilitates external clock synchronization using a GPS receiver.

14.5.4.2 Periodic Exchange of State Messages

TT Ethernet supports two types of communication, namely, standard event-triggered Ethernet traffic and the time-triggered traffic that is temporally guaranteed. For the time-triggered traffic, a static or dynamic scheduler has to define the conflict-free periods of the messages. The scheduler ensures that no conflict between TT Ethernet messages occurs. If an event-triggered Ethernet message comes into conflict with a time-triggered Ethernet message, then the TT Ethernet switch [SGAK06] preempts the transmission of the event-triggered message. After the completion of the transmission of the TT Ethernet message, the switch autonomously retransmit the preempted event-triggered Ethernet message.

Based on the time format, TT Ethernet restricts the period durations of time-triggered messages to the positive and negative powers of two of the second. Consequently, TT Ethernet supports up to 64 different period durations corresponding to the 64 different bits in the time format. The bit in the binary time format that specifies the period is denoted as the period bit. The phase of a message is specified with 12 bits to the right of the period bit.

14.5.4.3 Fault Isolation Mechanisms

TT Ethernet supports two configurations, namely, a non-fault-tolerant and a fault-tolerant configuration. The non-fault-tolerant configuration uses a single switch, whereas the fault-tolerant configuration requires two independent switches each of which is monitored by a guardian. The guardian observes the behavior of the switch and disables the outputs of the switch in case an error is detected. The error detection performed by the switch includes the monitoring of the switch delays, the compliance of the switch to the precomputed message schedule, and a semantic analysis of the time-triggered message header. The guardian has its own fault-tolerant clock synchronization algorithm and receives all time-triggered messages from the switch. The time-triggered message header in the fault-tolerant configuration is extended to contain a membership vector, the period counter, and a signature of the scheduler.

14.5.4.4 Diagnostic Services

In the fault-tolerant configuration, TT Ethernet provides a membership vector that is calculated according to the same algorithm as in TTP [TTT02b]. In addition, the guardian of the fault-tolerant configuration produces diagnostic messages to inform about detected errors [KAGS05].

14.5.4.5 Commercial or Prototypical Components

Prototypes of TT Ethernet are available as hardware- and software-based implementations. A hardware-based implementation is described in Ref. [SA07]. A VHDL module has been developed, which can be applied in System On a Programmable Chip systems. In addition, the VHDL module has been applied in a Cardbus compliant PCMCIA card to provide TT Ethernet support for devices equipped with a Cardbus interface.

In addition, a software implementation of a TT Ethernet controller has been realized [GASK06]. This implementation uses a Soekris 4801 www.soekris.com board, which incorporates a 266 MHz 568 class processor, 256 Mbyte RAM, and 3 Ethernet interfaces (see Figure 14.4). The TT Ethernet controller was implemented using the Linux-operating system with the RTAI extension.

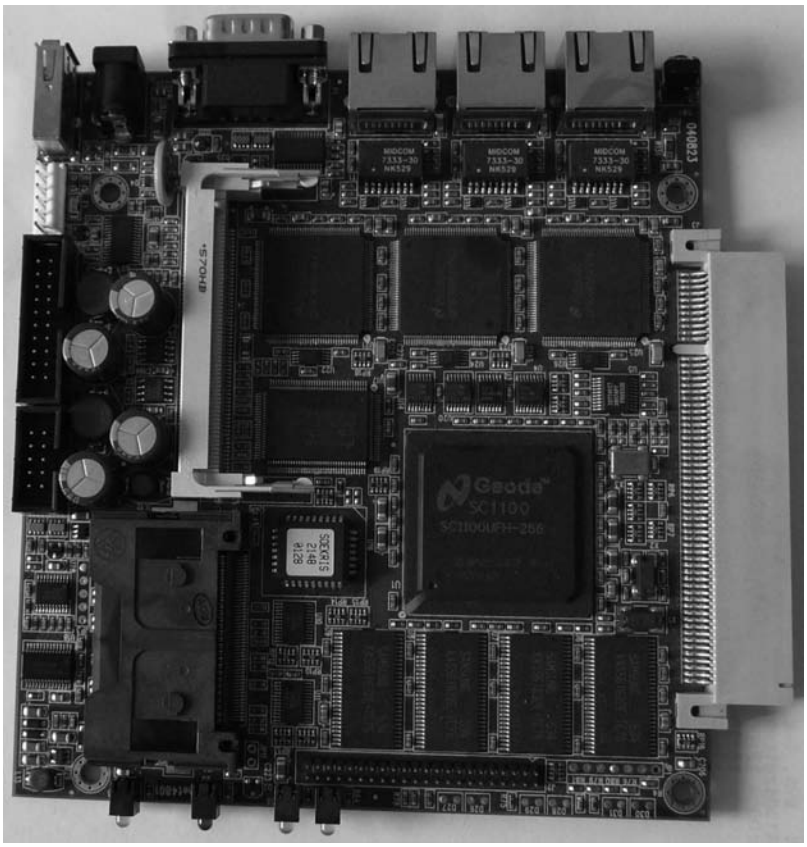


FIGURE 14.4 Time-triggered ethernet node based on Soekris 4801 in the software-based implementation.

References

[ALR00] A. Avizienis, J. Laprie, and B. Randell. Fundamental concepts of dependability. In *Proceedings of the ISW 2000. 34th Information Survivability Workshop*, pp. 7–12. IEEE, Boston, MA, 2000.

[ASBT03] A. Ademaj, H. Sivencrona, G. Bauer, and J. Torin. Evaluation of fault handling of the time-triggered architecture with bus and star topology. In *Proceedings of the International Conference on Dependable Systems and Networks*, pp. 123–132. Vienna University of Technology, Real-Time Systems Group, June 2003.

[Atm08a] T89C51CC01—Enhanced 8-bit microcontroller with can controller and flash memory. Technical report, Atmel Corporation, 2008.

[Atm08b] T89C51CC02—Enhanced 8-bit microcontroller with CAN controller and flash. Technical report, Atmel Corporation, 2008.

[AUT06] AUTOSAR GbR. *AUTOSAR—Technical Overview V2.0.1*, June 2006.

[BBD+00] D. Beal, E. Bianchi, L. Dozio, S. Hughes, P. Mantegazza, and S. Papacharalambous. RTAI: Real-time application interface. *Linux Journal*, Issue 72, Article No. 10, April 2000.

[BCV91] R.W. Butler, J.L. Caldwell, and B.L. Di Vito. Design strategy for a formally verified reliable computing platform. In *Proceedings of the 6th Annual Conference on Systems Integrity, Software Safety and Process Security*, pp. 125–133, Gaithersburg, MD, June 1991.

[BK00] G. Bauer and H. Kopetz. Transparent redundancy in the time-triggered architecture. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2000)*, pp. 5–13, New York, June 2000.

[BKAS] C. Bergenhem, J. Karlsson, C. Archer, and A. Sjöblom. Implementation results of a configurable membership protocol for active safety systems. In *Proceedings of the 12th Pacific Rim International Symposium on Dependable Computing (PRDC'06)*, Riverside, CA, 2006.

[BKS03] G. Bauer, H. Kopetz, and W. Steiner. The central guardian approach to enforce fault isolation in a time-triggered system. In *Proceedings of the 6th International Symposium on Autonomous Decentralized Systems (ISADS 2003)*, pp. 37–44, Pisa, Italy, April 2003.

[Bos91] Robert Bosch Gmbh, Stuttgart, Germany. *CAN Specification, Version 2.0*, 1991.

[Bos06] Automotive electronics—Product information TTCAN IP module. Technical report, Bosch, 2006.

[BP00] G. Bauer and M. Paulitsch. An investigation of membership and clique avoidance in TTP/C. In *Proceedings of 19th IEEE Symposium on Reliable Distributed Systems*, pp. 118–124, Nürnberg, Germany, October 2000.

[Bre01] E. Bretz. By-wire cars turn the corner. *IEEE Spectrum*, 38(4):68–73, April 2001.

[Car82] W.C. Carter. A time for reflection. In *Proceedings of the 8th IEEE International Symposium on Fault tolerand Computing (FTCS-8)*, p. 41, Santa Monica, CA, June 1982.

[CASD85] F. Cristian, H. Aghali, R. Strong, and D. Dolev. Atomic broadcast: From simple message diffusion to byzantine agreement. In *Proceedings of 15th International Symposium on Fault-Tolerant Computing (FTCS-15)*, pp. 200–206, IEEE Computer Society Press, Ann Arbor, MI, 1985.

[EHK+05] W. Elmenreich, W. Haidinger, H. Kopetz, T. Losert, R. Obermaisser, M. Paulitsch, and P. Peti. A standard for real-time smart transducer interface. *Computer Standards & Interfaces*, 28(6):613–624, 2006.

[Elm06] W. Elmenreich. Time-triggered smart transducer networks. *IEEE Transactions on Industrial Informatics,* 2(3):613–624, 2006.

[GASK06] P. Grillinger, A. Ademaj, K. Steinhammer, and H. Kopetz. Software implementation of time-triggered Ethernet controller. In *Workshop on Factory Communication Systems—WFCS 2006*, pp. 145–150, Torino, Italy, June 2006. ISBN 1-4244-0379-0.

[GP96] A. Galleni and D. Powell. Consensus and membership in synchronous and asynchronous distributed systems. Technical Report 96104, LAAS, April 1996.

[Gua05] FlexRay Consortium. BMW AG, DaimlerChrysler AG, General Motors Corporation, Freescale GmbH, Philips GmbH, Robert Bosch GmbH, and Volkswagen AG. *Node-Local Bus Guardian Specification Version 2.0.9*, December 2005.

[Inf03] TC1130—32-bit single-chip microcontroller data sheet. Technical report, Infineon Technologies, 2003.

[Int93] Int. Standardization Organisation, ISO 11898. *Road Vehicles—Interchange of Digital Information—Controller Area Network (CAN) for High-Speed Communication*, 1993.

[KAGS05] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer. The time-triggered ethernet (TTE) design. *Proceedings of the 8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*, Seattle, WA, May 2005.

[KBP01] H. Kopetz, G. Bauer, and S. Poledna. Tolerating arbitrary node failures in the time-triggered architecture. In *Proceedings of the SAE 2001 World Congress*, Detroit, MI, March 2001.

[Kel03] U. Kelling. MultiCAN—A step to CAN and TTCAN. In *Proceedings of the 9th International CAN Conference*, Munich, Germany, 2003.

[KGR91] H. Kopetz, G. Grünsteidl, and J. Reisinger. Fault-tolerant membership service in a synchronous distributed real-time system. *Dependable Computing for Critical Applications* (A. Avizienis and J. C. Laprie, eds.), pp. 411–429, Springer-Verlag, Wien, Austria, 1991.

[KO87] H. Kopetz and W. Ochsenreiter. Clock synchronization in distributed real-time systems. *IEEE Transactions on Computers*, 36(8):933–940, 1987.

[KO02] H. Kopetz and R. Obermaisser. Temporal composability. *Computing & Control Engineering Journal*, 13:156–162, August 2002.

[Kop92] H. Kopetz. Sparse time versus dense time in distributed real-time systems. In *Proceedings of the 12th International Conference on Distributed Computing Systems*, Japan, June 1992.

[Kop95] H. Kopetz. Why time-triggered architectures will succeed in large hard real-time systems. In *Proceedings of the 5th IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems*, Cheju Island, Korea, August 1995.

[Kop97] H. Kopetz. *Real-Time Systems, Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, Boston, MA, Dordrecht, the Netherlands, London, 1997.

[Kop03] H. Kopetz. Fault containment and error detection in the time-triggered architecture. In *Proceedings of the International Symposium on Autonomous Decentralized Systems*, Pisa, Italy, April 2003.

[Kop04] H. Kopetz. The fault-hypothesis for the time-triggered architecture. In *IFIP Congress Topical Sessions*, pp. 221–234, Toulouse, France, 2004.

[Kop08] H. Kopetz. The complexity challenge in embedded system design. In *Proceedings of the 11th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, Orlando, FL, 2008.

[Lap92] J.C. Laprie. Dependability: Basic concepts and terminology in English, French, German, Italian and Japanese, *Dependable Computing and Fault Tolerance*, p. 265, Springer Verlag, Vienna, Austria, 1992.

[LH94] J.H. Lala and R.E. Harper. Architectural principles for safety-critical real-time applications. *Proceedings of the IEEE*, 82(1):25–40, January 1994.

[LL84] J. Lundelius and N. Lynch. A new fault-tolerant algorithm for clock synchronization. In *Proceedings of the 3rd Annual ACM Symposium on Principles of Distributed Computing*, pp. 75–88. ACM Press, Providence, RI, 1984.

[Mac88] D.A. Mackall. Development and flight test experiences with a flight-crucial digital control system, technical paper 2857. Technical report, NASA Ames Research, 1988.

[MFH+02] B. Müller, T. Führer, F. Hartwich, R. Hugel, and H. Weiler. Fault tolerant TTCAN networks. In *Proceedings of the 8th International CAN Conference (iCC)*, Las Vegas, NV, 2002.

[OKS07] R. Obermaisser, H. Kraut, and C. Salloum. A transient-resilient system-on-a-chip architecture with support for on-chip and off-chip tmr. In *Proceedings of the 7th European Dependable Computing Conference 2008 (EDCC)*, Kaunas, Lithuania, 2007.

[Pit02] S. Pitzek. Description mechanisms supporting the configuration and management of TTP/A fieldbus systems. Master's thesis, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 3/3/182-1, 1040 Vienna, Austria, 2002.

[Pol96] S. Poledna. *Fault-Tolerant Real-Time Systems: The Problem of Replica Determinism*. Kluwer Academic Publishers, Norwell, MA, 1996.

[PSF99] H. Pfeifer, D. Schwier, and F.W. von Henke. Formal verification for time-triggered clock synchronization. In *Proceedings of 7th IFIP Working Conference on Dependable Computing for Critical Applications*, pp. 207–226, San Jose, CA, November 1999. Fakultat fur Inf., Ulm Univ.

[Rus99] J. Rushby. Partitioning for avionics architectures: Requirements, mechanisms, and assurance. NASA Contractor Report CR-1999-209347, NASA Langley Research Center, June 1999. Also to be issued by the FAA.

[Rus01] J. Rushby. Bus architectures for safety-critical embedded systems. *Proceedings of the 1st Workshop on Embedded Software (EMSOFT 2001)*, vol. 2211 of Lecture Notes in Computer Science (T. Henzinger and C. Kirsch, eds.), pp. 306–323, Springer-Verlag, Lake Tahoe, CA, October 2001.

[SA07] K. Steinhammer and A. Ademaj. Hardware implementation of the time-triggered ethernet controller. *International Embedded Systems Symposium (IESS) 2007*, Irvine, CA, June 2007.

[SGAK06] K. Steinhammer, P. Grillinger, A. Ademaj, and H. Kopetz. A time-triggered ethernet (TTE) switch. In *Proceedings of the Design, Automation and Test in Europe*, Munich, Germany, March 2006.

[Tan99] TTPos—the time-triggered and fault-tolerant RTOS. In *Real-Time Magazine 99-4*, 1999.

[TBW+00] T. Führer, B. Müller, W. Dieterle, F. Hartwich, and R. Hugel. Time-triggered CAN-TTCAN: Time-triggered communication on CAN. In *Proceedings of the 6th International CAN Conference (ICC6)*, Torino, Italy, 2000.

[Tem98] C. Temple. Avoiding the babbling-idiot failure in a time-triggered communication system. In *Proceedings of the Symposium on Fault-Tolerant Computing*, pp. 218–227, Munich, Germany, 1998.

[Tro02] C. Troedhandl. Architectural requirements for ttp/a nodes. Master's thesis, Technische Universitat Wien, Real-Time Systems Group, Vienna, Austria, 2002.

[TTT02a] Ttp powernode—a ttp development board for the time-triggered architecture with the as8202nf ttp network controller v2.01. Technical report, TTTech Computertechnik AG, 2002. www.tttech.com.

[TTT02b] TTTech Computertechnik AG, Schönbrunner Strasse 7, A-1040 Vienna, Austria. *Time-Triggered Protocol TTP/C—High Level Specification Document*, July 2002.

[TTT02c] TTTech Computertechnik AG, Schönbrunner Strasse 7, A-1040 Vienna, Austria. *TTP Monitoring Node—A TTP Development Board for the Time-Triggered Architecture*, March 2002.

15

Controller Area Networks for Embedded Systems

15.1 Introduction 15-1
CAN in Automotive Applications • CAN in Automation and Embedded Systems
15.2 CAN Protocol Basics 15-3
Physical Layer • Frame Format • Medium Access Technique • Error Management • Fault Confinement • Communication Services • Implementation
15.3 Schedulability Analysis of CAN Networks 15-14
Communication Requirements • Schedulability Analysis • Usage
15.4 Considerations about CAN 15-18
Advantages • Drawbacks • Solutions
15.5 Time-Triggered CAN 15-21
Main Features • Protocol Specification • Implementation
15.6 CANopen 15-25
CANopen Basics • Object Dictionary • Process Data Objects • Service Data Objects • Other Objects • Network Management
15.7 CANopen Device Profile for Generic I/O Modules ... 15-34
Device Definition • Device Behavior
References 15-37

Gianluca Cena
National Research Council

Adriano Valenzano
National Research Council

15.1 Introduction

Controller area network (CAN) is currently the leading serial bus for embedded control systems. More than two billion CAN nodes have been sold since the protocol was first introduced.

The history of CAN started more than 20 years ago. At the beginning of the 1980s a group of engineers at Bosch GmbH was looking for a serial bus system suitable for use in passenger cars. The most popular solutions adopted at that time were considered inadequate for the needs of most automotive applications. The bus system, in fact, had to provide a number of new features that could hardly be found in the already existing fieldbus architectures. The design of the new proposal also involved several academic partners and had the support of Intel, as the potential main semiconductor producer.

The new communication protocol was presented officially in 1986 with the name of "Automotive Serial Controller Area Network" at the SAE congress held in Detroit. It was based on a multi-master access scheme to the shared medium that resembled the well-known carrier sense

multiple access (CSMA) approach. The peculiar aspect, however, was that CAN adopted a new distributed nondestructive arbitration mechanism to solve contentions on the bus by means of priorities implicitly assigned to the colliding messages. Moreover, the protocol specifications also included a number of error detection and management mechanisms to enhance the fault tolerance of the whole system.

In the following years both Intel and Philips started to produce controller chips for CAN following two different philosophies. The Intel solution (often referred to as FullCAN in the literature) required less host CPU power, since most of the communication and network management functions were carried out directly by the network controller. Instead, the Philips solution (BasicCAN) was simpler but imposed a higher load on the processor used to interface the CAN controller. Since the mid-1990s more than 15 semiconductor vendors including Siemens, Motorola, and NEC have been producing and shipping millions of CAN chips mainly to car manufacturers such as Mercedes-Benz, Volvo, Saab, Volkswagen, BMW, Renault, and Fiat.

The Bosch specification (CAN version 2.0) was submitted for international standardization at the beginning of the 1990s. The proposal was approved and published as the ISO 11989 standard at the end of 1993 and contained the description of the medium access protocol and the physical layer architecture. In 1995 an addendum to ISO 11898 was approved to describe the extended format for message identifiers.

The CAN specification has recently been revised and reorganized, and has been split into several separate parts: part 1 [ISO1] defines the CAN data-link protocol, whereas part 2 [ISO2] and part 3 [ISO3] specify the high-speed (nonfault-tolerant) and low-speed (fault-tolerant) physical layers, respectively. Part 4 [ISO4] specifies the protocol for time-triggered communication (TTCAN). More recently, part 5 [ISO5] was added that deals with high-speed medium access units with low-power mode.

15.1.1 CAN in Automotive Applications

The main use of CAN has been (and still is) in the automotive industry. Here, CAN is adopted as in-vehicle network for engine management, comfort electronics (e.g., doors control, air conditioning, lightning, and so on) as well as for some infotainment (information and entertainment) functions. Currently, it is also being used as on-vehicle network for special-purpose cars, such as, for example, police cars and taxis.

The different CAN networks in a vehicle are connected through gateways. In many cases, this functionality is implemented in the dashboard, which may be equipped with a local CAN network for connecting displays and instruments.

High-speed CAN networks (e.g., 500 kbps), that comply with the ISO 11898-2 physical layer, are used by a vast majority of car makers in power-engine systems. In addition, most passenger cars produced in Europe are equipped with CAN-based multiplex networks that connect doors and roof control units as well as lighting and seat control units. These networks are often based on either the ISO 11898-3 fault-tolerant physical layer or the ISO 11898-5 high-speed physical layer with low-power functionality.

Sometimes, also infotainment devices are interconnected by means of CAN. Besides proprietary solutions, the Society of Automotive Engineers (SAE) defined the IDB-C application profile in the ITS Data Bus series of specifications (SAE J2366), which is based on high-speed CAN (29 bit identifiers) with a bit-rate of 250 kbps. Finally, a CAN-based diagnostic interface is provided by several passenger cars for connecting diagnostics tools to in-vehicle networks. Several international standards are available in this case, namely ISO 16765 (Diagnostics on CAN including Keyword 2000 over CAN), ISO 14229-1 (Unified Diagnostic Services), and ISO 15031 (on-board diagnostic standard).

Electronic control units (ECUs) for vehicles (e.g., passenger cars, trucks, and buses) are in most cases original equipment manufacturer (OEM)-specific. This means they comply with the

specifications dictated by the OEM for what concerns both the physical (e.g., bit-timing) and higher-layer protocols. While trucks and buses mostly rely on J1939-based specifications, proprietary higher-layer protocols and profiles are usually adopted in passenger cars. As an exception, CAN-based diagnostics interfaces to scanner tools usually comply with ISO standards.

15.1.2 CAN in Automation and Embedded Systems

Even though it was conceived explicitly for vehicle applications, at the beginning of the 1990s CAN began to be adopted in different scenarios, too. The standard documents provided satisfactory specifications for the lower communication layers but did not offer guidelines or recommendations for the upper part of the Open Systems Interconnection (OSI) protocol stack, in general, and for the application layer in particular. This is why the earlier applications of CAN outside the automotive scenario (i.e., textile machines, medical systems, and so on) adopted "ad hoc" monolithic solutions. The "CAN in Automation" (CiA) users' group, founded in 1992, was originally concerned with the specification of a "standard" CAN application layer (CAL). This effort led to the development of the general-purpose CAL specification. CAL was intended to fill the gap between the distributed application processes and the underlying communication support but, in practice, it was not successful, the main reason being that, as CAL is really application-independent, each user had to develop suitable profiles for his/her specific application fields.

In the same years, Allen-Bradley and Honeywell started a joint distributed control project based on CAN. Although the project was abandoned a few years later, Allen Bradley and Honeywell continued their works separately and focused on the higher protocol layers. The results of those activities were the Allen-Bradley DeviceNet solution and the honeywell smart distributed system (SDS). For a number of reasons SDS remained, in practice, a solution internal to Honeywell Microswitch, while the development and maintenance activities for DeviceNet were soon switched to the Open DeviceNet Vendor Association (ODVA). Thereafter, DeviceNet has been widely adopted in the United States in a number of factory automation areas, so becoming a serious competitor to widespread solutions such as PROFIBUS-DP and INTERBUS.

Besides DeviceNet and SDS other significant initiatives were focused on CAN and its application scenarios. CANopen was conceived in the framework of the European Esprit project ASPIC by a consortium led once again by Bosch GmbH. The purpose of CANopen was to define a profile based on CAL, which could support communications inside production cells. The original CANopen specifications were further refined by CiA and released in 1995.

Later on, CANopen, DeviceNet, and SDS became European standards, CANopen as an embedded network for machine-distributed control and the other two for factory and process automation. In particular, they are defined in standard EN 50325 [EN1], where part 2 is about DeviceNet [EN2], part 3 deals with SDS [EN3], and part 4 describes CANopen [EN4].

15.2 CAN Protocol Basics

The CAN protocol architecture is structured according to the well-known layered approach foreseen by the ISO OSI model. However, as in most currently existing networks conceived for use at the field level in the automated manufacturing environments, only few layers have been considered in its protocol stack. This is to make implementations more efficient and inexpensive. Few protocol layers, in fact, imply reduced processing delays when receiving and transmitting messages, as well as simpler communication software.

CAN specifications [ISO1] [ISO2] [ISO3], in particular, include only the physical and data-link layers, as depicted in Figure 15.1. The physical layer is aimed at managing the effective transmission of data over the communication support, and tackles the mechanical and electrical aspects. Bit-timing and synchronization, in particular, belong to this layer.

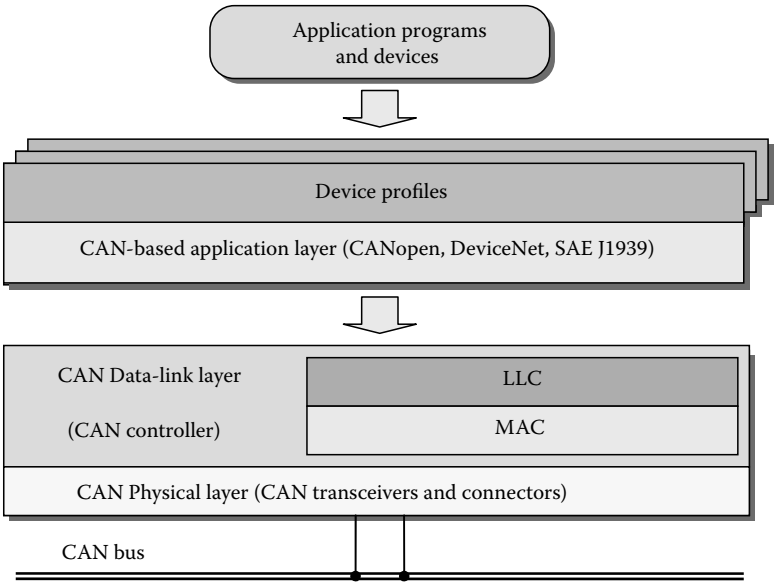


FIGURE 15.1 CAN protocol stack.

The data-link layer, instead, is split into two separate sublayers, namely the medium access control (MAC) and the logical link control (LLC). The purpose of the MAC entity is basically to manage the access to the shared transmission support by providing a mechanism aimed at coordinating the use of the bus, so as to avoid unmanageable collisions. The functions of the MAC sublayer include frame encoding and decoding, arbitration, error checking and signaling, and also fault confinement. The LLC sublayer, instead, offers the user (i.e., application programs running in the upper layers) a proper interface, which is characterized by a well-defined set of communication services, in addition to the ability to decide whether or not an incoming message is relevant to the node.

It is worth noting that the CAN specification is very flexible concerning both the implementation of the LLC services and the choice of the physical support, whereas modifications to the behavior of the MAC sublayer are not admissible.

As mentioned before, unlike most fieldbus networks the CAN specification does not include any native application layer. However, a number of such protocols exist that rely on CAN and ease the design and the implementation of complex CAN systems.

15.2.1 Physical Layer

The features of the physical layer of CAN that are valid for any system, such as those related to the physical signaling, are described in the ISO 11898-1 document [ISO1]. Instead, medium access units (i.e., transceivers) are defined in separate documents, e.g., ISO 11898-2 [ISO2] and ISO 11898-3 [ISO3] for high- and low-speed communications, respectively. The definition of the medium interface (i.e., connectors) is usually covered in other documents.

15.2.1.1 Network Topology

CAN networks are based on a shared bus topology. Buses have to be terminated with resistors at each end (the recommended nominal impedance is 120 Ω), so as to suppress signal reflections. For the same reason standard documents state that the topology of a CAN network should be as close as

possible to a single line. Stubs are permitted for connecting devices to the bus, but their length should be as short as possible. For example, at 1 Mbps the length of a stub must be shorter than 30 cm.

Several kinds of transmission media can be used:

- Two-wire bus, which enables differential signal transmissions and ensures reliable communications. In this case, shielded twisted pair can be used to further enhance the immunity to electromagnetic interferences.
- Single-wire bus, a simpler and cheaper solution that features lower immunity to interferences and is mainly suitable for use in automotive applications.
- Optical transmission medium, which ensures complete immunity to electromagnetic noise and can be used in hazardous environments. Fiber optics is often adopted to interconnect (through repeaters) different CAN subnetworks. This is done to cover plants that are spread over a large area.

Several bit rates are available for the network, the most adopted being in the range from 50 kbps to 1 Mbps (the latter value represents the maximum allowable bit rate according to the CAN specifications). The maximum extension of a CAN network depends directly on the bit rate. The exact relation between these two quantities also involves parameters such as delays introduced by transceivers and optocouplers. Generally speaking, the mathematical product between the length of the bus and the bit rate has to be approximately constant. For example, the maximum extension allowed for a 500 kbps network is about 100 m, and increases up to about 500 m when a bit rate of 125 kbps is considered.

Signal repeaters can be used to increase the network extension, especially when large plants have to be covered and the bit rate is low or medium. However, they introduce additional delays on the communication paths, hence the maximum distance between any two nodes is effectively shortened at high bit rates. Using repeaters also achieves topologies different from the bus (trees or combs, for example). In this case, a good design could increase the effective area which is covered by the network.

It is worth noting that, unlike other field networks such as, for example, PROFIBUS-PA, there is in general no cost-effective way in CAN to use the same wire for carrying both the signal and the power supply. However, an additional pair of wires can be provided inside the bus cable for power supply.

Curiously enough, connectors are not standardized by CAN specifications. Instead, several companion or higher-level application standards exist that define their own connectors and pin assignment. CiA DS102 [DS102], for example, foresees the use of a SUB-D9 connector, while DeviceNet and CANopen also suggest the use of either 5-pin mini-style, microstyle, or open-style connectors. In addition, these documents include recommendations for bus lines, cables, and standardized bit rates, which were not included in the original CAN specifications.

15.2.1.2 Bit Encoding and Synchronization

In CAN the electrical interface of a node to the bus is based on an open-collector-like scheme. As a consequence, the level on the bus can assume two complementary values, which are denoted symbolically as dominant and recessive. Usually, the dominant level corresponds to the logical value 0 whilst the recessive level is the same as the logical value 1.

CAN relies on the nonreturn to zero bit encoding, which features very high efficiency in that the synchronization information is not encoded separately from data. Bit synchronization in each node is achieved by means of a digital phase-locked loop, which extracts the timing information directly from the bit stream received from the bus. In particular, edges of the signal are used for synchronizing local clocks, so as to compensate tolerances and drifts of oscillators.

To provide a satisfactory degree of synchronization among nodes, the transmitted bit stream should include a sufficient number of edges. To do this CAN relies on the so-called bit stuffing technique. In practice, whenever 5 consecutive bits at the same value (either dominant or recessive) appear in the transmitted bit stream, the transmitting node inserts one additional stuff bit at

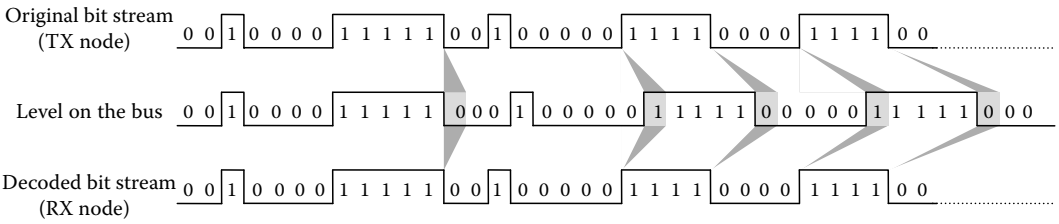


FIGURE 15.2 Bit stuffing technique.

the complementary value, as depicted in Figure 15.2. Stuff bits can be easily and safely removed by receiving nodes, so as to obtain the original stream of bits back.

From a theoretical point of view, the maximum number of stuff bits that may be added is one every 4 bits in the original frame, so the encoding efficiency can be as low as 80% (see, for example, the rightmost part of Figure 15.2, where the original bit stream alternates sequences of 4 consecutive bits at the dominant level followed by 4 bits at the recessive level). However, the influence of bit stuffing in real operating conditions is noticeably lower than the theoretical value mentioned above. Simulations show that, on average, only 2 to 4 stuff bits are effectively added to each frame, depending on the size of the identifier and data fields. Despite it is quite efficient, the bit stuffing technique has a drawback. In particular, the time taken to send a message over the bus is not fixed; instead it depends on the content of the message itself. This might cause unwanted jitters.

Not all the fields in a CAN frame are encoded according to the bit stuffing mechanism: in particular, it applies only to the initial part of the frames, from the start of frame (SOF) bit up to the cyclic redundancy check (CRC) sequence. Remaining fields, instead, are of fixed form and are never stuffed.

15.2.2 Frame Format

The CAN specification [ISO1] defines both a standard and an extended frame format. These formats mainly differ for the size of the identifier field and for some other bits in the arbitration field. In particular, the standard frame format (that is also known as CAN 2.0A format) defines an 11 bit identifier field, which means that up to 2048 different identifiers are available to applications running in the same network (many older CAN controllers, however, only support identifiers in the range from 0 to 2031). The extended frame format (identified as CAN 2.0B), instead, assigns 29 bits to the identifier, so that up to half a billion different objects could exist (in theory) in the same network. This is a fairly high value, which is virtually sufficient for any kind of application.

Using extended identifiers in a network that also includes 2.0A compliant CAN controllers, usually leads to unmanageable transmission errors, which effectively make the network unstable. Thus, a third category of CAN controllers was developed, known as 2.0B passive: they manage in a correct way the transmission and the reception of CAN 2.0A frames, whilst CAN 2.0B frames are simply ignored so that they do not hang the network.

It is worth noting that, in most practical cases, the number of different objects allowed by the standard frame format is more than adequate. Since standard CAN frames are shorter than extended frames (because of the shorter arbitration field), they enable higher communication efficiency (unless part of the payload is moved into the arbitration field). As a consequence they are adopted in most of the existing CAN systems, and also most of the CAN-based higher-layer protocols, such as CANopen and DeviceNet, basically rely on this format.

The CAN protocol foresees only four kinds of frames, namely data, remote, error and overload frames. Their format is described in detail in Sections 15.2.2.1 through 15.2.2.3.

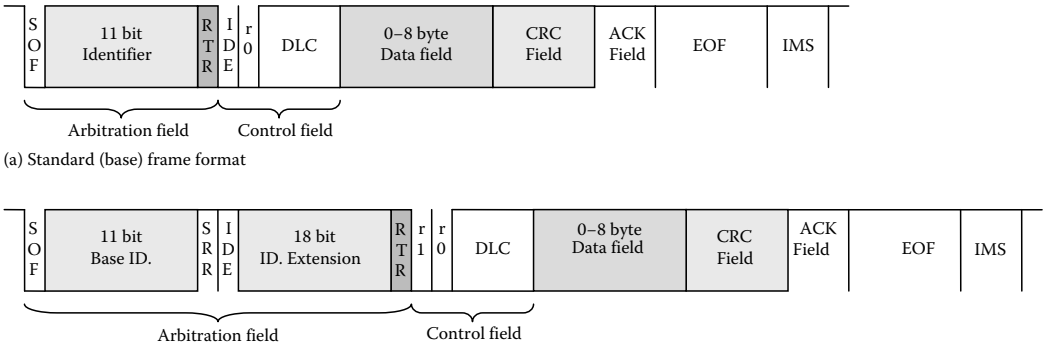


FIGURE 15.3 Format of data frames.

15.2.2.1 Data Frame

Data frames are used to send information over the network. Each CAN data frame begins with an SOF bit at the dominant level, as shown in Figure 15.3. Its role is to mark the beginning of the frame, as in serial transmissions carried out by means of conventional universal asynchronous receiver/transmitters (UARTs). The SOF bit is also used to synchronize the receiving nodes.

Immediately after the SOF bit the arbitration field follows, which includes both the identifier field and the remote transmission request (RTR) bit. As the name suggests, the identifier field identifies the content of the frame which is being exchanged uniquely on the whole network. The identifier is also used by the MAC sublayer to detect and manage the priority of the frame, which is used whenever a collision occurs (the lower the numerical value of the identifier, the higher the priority of the frame).

The identifier is sent starting from the most significant bit up to the least significant one. The size of the identifier is different for the standard (base) and the extended frames. In the latter case the identifier has been split into an 11 bit base identifier and an 18 bit identifier extension, so as to provide compatibility with the standard frame format.

The RTR bit is used to discriminate between data and remote frames. Since a dominant value of RTR denotes a data frame whilst a recessive value stands for a remote frame, a data frame has a higher priority than a remote frame having the same identifier.

Next to the arbitration field comes the control field. In the case of base frames, it includes the identifier extension (IDE) bit, which discriminates between base and extended frames, followed by the reserved bit *r0*. Instead, in extended frames the IDE bit effectively belongs to the arbitration field, as well as the substitute remote request (SRR) bit—a placeholder that is sent at recessive value to preserve the structure of the frames. In this case the IDE bit is followed by the identifier extension and then by the control field, that begins with the two reserved bits *r*1 and *r*0. After the reserved bits the data length code (DLC) follows, which specifies the length (in number of bytes encoded on 4 bits) of the data field. Since the IDE bit is dominant in base format frames whilst it is recessive in the extended ones, standard frames have precedence over their extended counterparts when the same base identifier is considered.

Reserved bits *r*0 and *r*1 must be sent by the transmitting node at the dominant value. Receivers, however, will ignore the value of these bits. For the DLC field, values ranging from 0 to 8 are allowed. According to the last specification, higher values (from 9 to 15) can be used for application-specific purposes. In this case, however, the length of the data field is meant to be 8.

The data field is used to store the effective payload of the frame. In order to ensure a high degree of responsiveness and minimize the priority inversion phenomenon, the size of the data field is limited to 8 bytes at most.

The CRC and acknowledgment fields are placed after the data field. CRC sequence is encoded on 15 bits, which is followed by a CRC delimiter at the recessive value. The kind of CRC adopted in CAN is particularly suitable to cover short frames (i.e., counting less than 127 bits). The acknowledgment field, instead, is made up of two bits, that is to say the ACK slot followed by the ACK delimiter. Both of them are sent at the recessive level by the transmitter. The ACK slot, however, is overwritten with a dominant value by each node that has received the frame correctly (i.e., no error was detected up to the ACK field). It is worth noting that, in this way, the ACK slot is actually surrounded by two bits at recessive level, that is to say the CRC and ACK delimiters. By means of the ACK bit, the transmitting node is enabled to discover whether or not at least one node in the network has received the frame correctly.

The frame is closed by the end of frame (EOF) field, made up of 7 recessive bits, which notifies all the nodes the end of an error-free transmission. In particular, the transmitting node assumes that the frame has been exchanged correctly if no error is detected until the last bit of the EOF field, whilst in the case of receivers the frame is valid if no error is detected until the sixth bit of EOF.

Different frames are interleaved by the intermission (IMS) gap, which consists of 3 recessive bits and effectively separates consecutive frames exchanged on the bus.

15.2.2.2 Remote Frames

Remote frames are very similar to data frames. The only difference is that they carry no data (i.e., the data field is not present in this case). They are used to request that a given message be sent on the network by a remote node. It is worth noting that the requesting node does not know who is the producer of the relevant information. It is up to receivers to discover the one that has to reply.

Indeed, the DLC field in remote frames is not used by the CAN protocol. However, it should be set to the same value as in the corresponding data frame, so as to cope with situations where several nodes send remote requests with the same identifier at the same time (this is legal in a CAN network). In that case, it is necessary for the different requests to be perfectly identical, so that they can overlap totally in the case of a collision.

It should be noted that, because of the way the RTR bit is encoded, if a request is made for an object at the same time as the transmission of that object is started by the producer, the contention is resolved in favor of the data frame.

15.2.2.3 Error and Overload Frames

Error frames are used to notify nodes in the network that an error has occurred. They consist of two fields: error flag and error delimiter. There are two kinds of error flags: active error flags are made up of 6 dominant bits, while passive error flags consist of 6 recessive bits. An active error flag violates the bit stuffing rules or the fixed-format parts of the frame which is currently being exchanged; hence it enforces an error condition that is detected by all the stations connected to the network. Each node, which detects an error condition, starts transmitting an error flag on its own. In this way, there can be from 6 to 12 dominant bits on the bus, as a consequence of the transmission of an error flag.

The error delimiter, instead, is made up of 8 recessive bits. After the transmission of an error flag, each node starts sending recessive bits and, at the same time, it monitors the bus level until a recessive bit is detected. At this point the node sends 7 more recessive bits, hence completing the error delimiter.

Overload frames can be used by slower receivers to slow down operations on the network. This is done by adding extra delays between consecutive data and/or remote frames. The overload frame format is very similar to error frames. In particular, it is made up of an overload flag followed by an overload delimiter. It is worth noting that today's CAN controllers are very fast, so overload frames have become almost useless.

15.2.3 Medium Access Technique

The MAC mechanism, which CAN relies on, is basically the CSMA scheme. When no frame is being exchanged the network is idle and the level on the bus is recessive. Before transmitting a frame, nodes have to sense the state of the network. If the network is found idle the frame transmission begins immediately, otherwise the node must wait for the end of the current frame transmission. Each frame starts with the SOF bit at the dominant level, which informs all the other nodes that the network has switched to the busy state.

Although a very unlikely event, two or more nodes might start sending their frames exactly at the same time. This may actually occur, because propagation delays on the bus—even though very small—are anyway greater than zero. Thus, one node might start its transmission while the SOF bit of another frame is already traveling on the bus. In this case, a collision occurs. In CSMA networks that are based on collision detection such as, for example, nonswitched Ethernet, this unavoidably leads to the corruption of all frames involved, which means they have to be retransmitted. The consequence is a waste of time and a net decrease of the available system bandwidth. In high load conditions, this may lead to congestion: when the number of collisions is so high that the net throughput on the Ethernet network falls below the arrival rate, the network is stalled.

Unlike Ethernet, CAN is able to resolve contentions in a deterministic way, so that neither time nor bandwidth is wasted. Therefore, congestion conditions can no longer occur and all the theoretical system bandwidth is effectively available for communications.

For the sake of truth, it should be said that contentions in CAN occur more often than one may think. In fact, when a node that has a frame to transmit finds the bus busy or loses the contention, it waits for the end of the current frame exchange and, immediately after the intermission has elapsed, it starts transmitting. Here, the node may compete with other nodes for which—in the meanwhile—a transmission request has been issued. In this case, different nodes synchronize on the falling edge of the first SOF bit which is sensed on the network.

This implies that CAN effectively behaves like a network-wide distributed transmission queue where messages are selected for transmission according to a priority order.

15.2.3.1 Bus Arbitration

The most distinctive feature of the medium access technique of CAN is the ability to resolve any collision that may occur on the bus in a deterministic way. In turn, this is made possible by the arbitration mechanism, which effectively finds out the most urgent frame each time there is a contention for the bus.

The CAN arbitration scheme allows collisions to be resolved by stopping transmissions of all frames involved except the one which is assigned the highest priority (i.e., the lowest identifier). The arbitration technique exploits the peculiarity of the physical layer of CAN, which conceptually provides a wired-and connection scheme among all nodes. In particular, the level on the bus is dominant if at least one node is sending a dominant bit; vice versa, the level on the bus is recessive if all nodes are transmitting recessive bits.

By means of the so-called binary countdown technique each node transmits the message identifier serially on the bus immediately after the SOF bit, starting from the most significant bit. When transmitting, each node compares the level sensed on the bus to the value of the bit which is being written out. If the node is transmitting a recessive value whereas the level on the bus is dominant, the node understands it has lost the contention and withdraws immediately. In particular, it ceases transmitting and sets its output port to the recessive level so as not to interfere with other contending nodes. At the same time it switches to the receiving state so as to read the incoming (winning) frame.

The binary countdown techniques ensures that, in the case of a collision, all nodes that are sending lower priority frames will abort their transmissions by the end of the arbitration field, except for the

one which is sending the frame with the highest priority (it should be noted that the winning node does not even realize that a collision has occurred). This implies that no two nodes in a CAN network can be transmitting messages concerning the same object (that is to say, characterized by the same identifier) at the same time. If this is not the case, in fact, unmanageable collisions could take place which, in turn, cause transmission errors. Because of the automatic retransmission feature of CAN controllers, this would lead almost certainly to a burst of errors on the bus, which, in turn, causes the stations involved to be disconnected by the fault confinement mechanism. This implies that, in general, only one node can be the producer of each object.

One exception to this rule is given by frames that do not have a data field, such as, for example, remote frames. In this case, should a contention occur involving frames with the same identifier, they would overlap perfectly and hence no collision effectively would occur. The same is also true for data frames whose data field is not empty, provided that its content is the same for all frames sharing the same identifier. However, it makes no sense in general to send frames with a constant data field.

All nodes that lose a contention have to retry the transmission as soon as the exchange of the current (winning) frame ends. They all try to send their frames again immediately after the intermission is read on the bus. Here, a new collision may take place that could also involve frames sent by nodes for which a transmission request was issued while the bus was busy.

An example that shows the detailed behavior of the arbitration phase in CAN is depicted in Figure 15.4. There, 3 nodes (that have been indicated symbolically as A, B, and C) start transmitting a frame at the same time (maybe at the end of the intermission following the previous frame exchange over the bus). As soon as a node understands it has lost the contention, it switches its output level to the recessive value, so that it no longer interferes with the other transmitting nodes. This event takes place when bit ID_5 is sent for node A, while the same happens for node B at bit ID_2. Node C manages to send the whole identifier field, and then it can keep on transmitting the remaining part of the frame.

15.2.4 Error Management

One of the main requirements that were considered as fundamental in the definition of the CAN protocol was the need to have a highly robust communication system, i.e., a system which is able to detect most of transmission errors. Hence, particular care was taken in defining the support for error management.

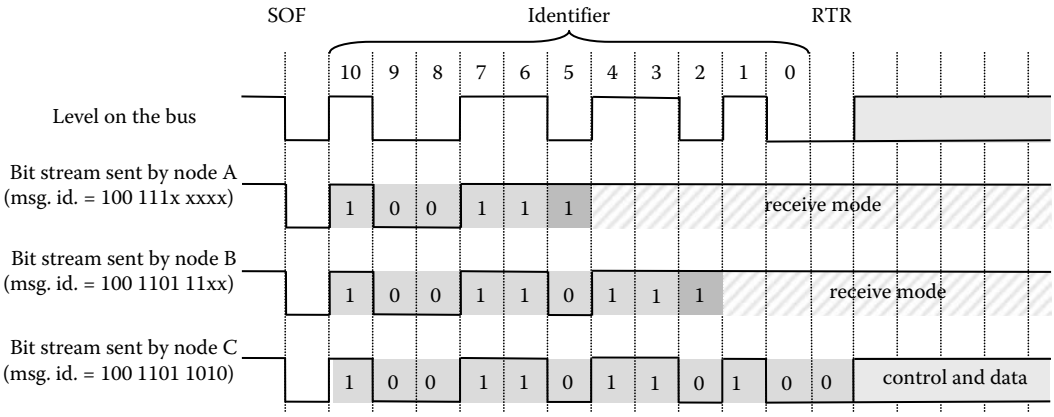

SOF
Identifier
RTR
10 9 8 7 6 5 4 3 2 1 0
Level on the bus
Bit stream sent by node A
(msg. id. = 100 111x xxxx)
1 0 0 1 1 1
receive mode
Bit stream sent by node B
(msg. id. = 100 1101 11xx)
1 0 0 1 1 0 1 1 1
receive mode
Bit stream sent by node C
(msg. id. = 100 1101 1010)
1 0 0 1 1 0 1 1 0 1 0 0
control and data

FIGURE 15.4 Arbitration phase in CAN.

The CAN specification foresees five different mechanisms to detect transmission errors:

1. *Cyclic redundancy check*: When transmitting a frame, the originating node appends a 15-bit-wide CRC to the end of the frame. Receiving nodes reevaluate the CRC to check if it matches the transmitted value. Generally speaking, the CRC used in CAN is able to discover up to 5 erroneous bits arbitrarily distributed in the frame and also errors bursts including up to 15 bits.
2. *Frame check*: Fixed-format fields in the received frames can be easily tested against their expected values. For example, the CRC and ACK delimiters as well as the EOF field have to be at the recessive level. If one or more illegal bits are detected, a form error is generated.
3. *Acknowledgment check*: The transmitting node verifies whether the ACK bit in the frame that is being sent has been set to the dominant value. If this does not occur, an acknowledgment error is issued. Such a kind of check is useful to discover whether or not there are other active nodes in the network (i.e., if the link of the transmitting node is not broken).
4. *Bit monitoring*: Each transmitting node compares the level on the bus to the value of the bit which is being written. Should a mismatch occur, an error is generated. It is worth noting that the same does not hold for the arbitration field and the acknowledgment slot. Such an error check is very effective to detect local errors that may occur in the transmitting nodes.
5. *Bit stuffing*: Each node verifies whether the bit stuffing rules have been violated in the portion of the frames from the SOF bit up to the CRC sequence. In the case 6 bits of identical value are read from the bus, an error is generated.

The residual probability that a corrupted message goes undetected in a CAN network—under realistic operating conditions—has been evaluated to be about 4.7×10^{-11} times the error rate or less.

15.2.5 Fault Confinement

So as to prevent a node that is not operating properly from sending corrupted frames repeatedly, thus blocking the whole network, a fault confinement mechanism has been included in the CAN specification. The fault confinement unit supervises the correct operation of the related MAC sublayer and, should the node become defective, it disconnects that node from the bus.

The fault confinement mechanism has been conceived so as to discriminate, as long as it is possible, between permanent failures and short disturbances which may cause bursts of errors on the bus. According to this mechanism, each node can be in one of the three following states:

- Error active
- Error passive
- Bus-off

Error active and error passive nodes take part in the communication in the same way. However, they react to error conditions differently. They send active error flags in the former case and passive error flags in the latter. This is because an error passive node has already experienced several errors, so it should avoid interfering with the network operations (a passive error flag, in fact, cannot corrupt the ongoing frame exchange).

The fault confinement unit uses two counters so as to track the behavior of the node with respect to the transmission errors: they are known as transmission error count (TEC) and receive error count (REC), respectively. The rules by which TEC and REC are managed are actually quite complex. However, they can be summarized as follows: each time an error is detected, the counters are increased by a given amount, whereas successful exchanges decrease them by one. Furthermore, the amount of the increase for the nodes which first detected the error is higher than the nodes that simply replied to the error flag. In this way it is very likely that counters of faulty nodes increase more

quickly than in nodes that are operating properly, even when sporadic errors are considered caused by electromagnetic noise.

When one of the error counters exceeds a first threshold (127), the node is switched to the error passive state, so as to try not to affect the network. It will return to the error active state as soon as counters fall below the above threshold. When a second threshold (255) is exceeded, instead, the node is switched to the bus-off state. At this point, it can no longer transmit any frame on the network, and it can be switched back to the error-active state only after it has been reset and reconfigured.

15.2.6 Communication Services

According to the ISO specification [ISO1], the LLC sublayer of CAN provides two communication services only: *L_DATA*, which is used to broadcast the value of a specific object over the network, and *L_REMOTE*, which is used to ask for the value of a specific object to be broadcast by its remote producer. From a practical point of view, these primitives are implemented directly in the h/w by all the currently available CAN controllers.

15.2.6.1 Model for Information Exchanges

Unlike most network protocols conceived for the use in automated manufacturing environments (which rely on node addressing) CAN adopts object addressing. In other words, messages are not tagged with the address of the destination and/or originating nodes. Instead, each piece of information that is exchanged over the network (often referred to as an object) is assigned a unique identifier, which denotes unambiguously the meaning of the object itself in the whole system.

This fact has important consequences on the way communications are carried out in CAN. In fact, identifying objects that are exchanged over the network according to their meaning rather than to the node they are intended for, allow implicit multicasting and makes it very easy for the control applications to manage interactions among devices according to the producer/consumer paradigm.

The exchange of information in CAN takes place according to the three phases shown in Figure 15.5:

1. Producer of a given piece of information encodes and transmits the related frame on the bus (the arbitration technique will transparently resolve any contention which could occur).

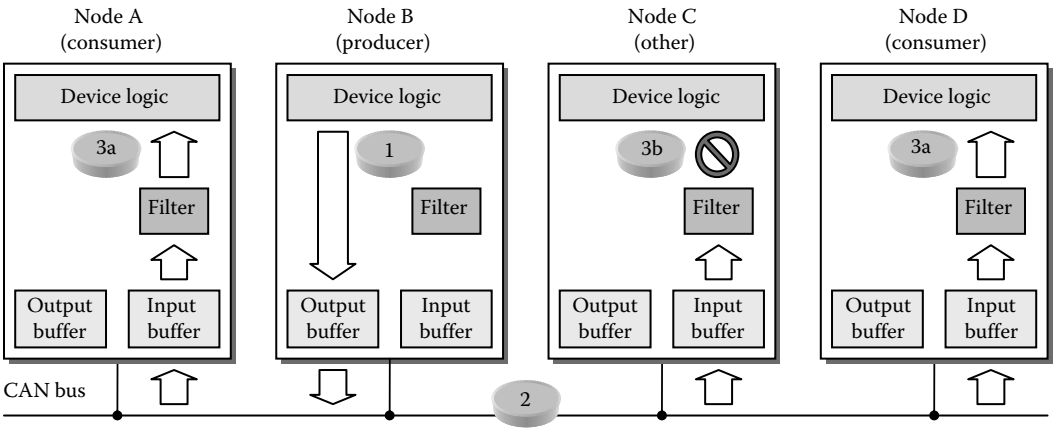


FIGURE 15.5 Producer/consumer model.

2. Because of the intrinsic broadcast nature of the bus, the frame is propagated all over the network, and every node reads its content in a local receive buffer.
3. Frame acceptance filtering (FAF) function in each node determines whether or not the information is relevant to the node itself. In the former case the frame is passed to the upper communication layers (from a practical point of view, this means that the CAN controller raises an interrupt to the local device logic, which will then read the value of the object); on the contrary, the frame is simply ignored and discarded.

In the sample data exchange depicted in Figure 15.5, node B is the producer of some kind of information, which is relevant to (i.e., consumed by) nodes A and D. Node C, instead, is not interested in such data, which are rejected by the filtering function (this is the default behavior of the FAF function).

15.2.6.2 Model for the Interaction of Devices

The access technique of CAN makes this kind of network particularly suitable for distributed systems that communicate according to the producer/consumer model. In this case, data frames are used by producer nodes to broadcast new values over the network, and each datum is identified unambiguously by means of its identifier. Unlike those networks based on the producer/consumer/arbiter model such as Factory Instrumentation Protocol (FIP), in CAN information is sent as soon as it becomes available from either the control applications or the controlled physical system (by means of sensors), without any need for the intervention of a centralized arbiter. This noticeably improves the responsiveness of the whole system.

CAN networks work equally well when they are used to interconnect devices in systems based on a more conventional master/slave communication model. In this case, the master can use remote frames to solicit some specific information to be remotely sent on the network. As a consequence, the information producer replies with a data frame carrying the relevant object.

It is worth noting that this kind of interaction is implemented in CAN in a fairly more flexible way than in conventional master/slave networks, such as, for example, PROFIBUS-DP. In CAN, in fact, the reply (data frame) does not need to follow the request (remote frame) immediately. In other words, the network is not kept busy while waiting for the device's reply. This allows the system bandwidth to be, in theory, fully available for applications. Furthermore, the reply containing the requested value is broadcast on the whole network, and hence it can be read by all interested nodes, besides the one which issued the remote request.

15.2.7 Implementation

According to the internal architecture, CAN controllers can be classified into two different categories, namely BasicCAN and FullCAN.

Conceptually, BasicCAN controllers are provided with one transmit and one receive buffer, as in conventional UARTs. The frame filtering function, in this case, is generally left to the application programs (i.e., it is under control of the host controller), even though some kind of filtering can be done by the controller. So as to avoid overrun conditions, a double-buffering scheme based on shadow receive buffers is usually provided, and this enables the reception of a new frame from the bus while the previous one is being read by the host controller. An example of controller based on the BasicCAN scheme is the PCA82C200 component by Philips. In more recent controllers, such as the Philips SJA 1000, a 64-byte receive FIFO queue is available.

FullCAN implementations, instead, foresee a number of internal buffers that can be configured to either receive or transmit some particular messages. In this case the filtering function is implemented directly in the CAN controller. When a new frame, which is of interest for the node, is received from the network, it is stored in the related buffer, where it can be subsequently read by the host controller. In general, new values simply overwrite the older ones, and this does not lead to overrun conditions

(the old value of a variable is simply replaced with a new one). The Intel 82526 and 82527 CAN controllers are based on the FullCAN architecture.

FullCAN controllers, in general, free the host controller of a number of activities, so they are considered to be more powerful than BasicCAN controllers. It is worth noting, however, that recent CAN controllers are somehow able to embed the operating principles of both architectures, so the classification mentioned above is progressively being superseded.

15.3 Schedulability Analysis of CAN Networks

From a theoretical point of view, the CAN protocol can be seen as a priority-based nonpreemptive distributed system. This means that several techniques are available in literature, most of which are derived from fixed-priority scheduling in real-time operating systems, to analyze timings in a CAN network. In particular, under well-defined (yet typical, for networked embedded systems) assumptions, it is possible to determine the maximum latency a message may experience before it is delivered to the destination. This, in turn, enables to assess whether or not a given set of messages exchanged over the network, as a consequence of the operation of a real-time distributed control application, is really schedulable (i.e., timing requirements are fulfilled).

From a general viewpoint, nothing can be said about communication latencies if messages are generated in a completely arbitrary way. However, if the message generation pattern is known in advance (e.g., each message is exchanged periodically), worst-case transmission times can be easily evaluated. This applies also when message generation processes on different nodes are not coordered in any way (i.e., different transmission patterns are displaced by arbitrary offsets).

The theory behind such an analysis is not trivial, and has been clearly introduced and thoroughly explained in several papers [TIN94a] [TIN94b] [TIN95]. In the following, a simplified approach, first appeared in 1994, will be described, that provides nevertheless useful results in real-world operating conditions, particularly for the networked embedded systems typically found in most automotive applications.

15.3.1 Communication Requirements

The first step for analyzing schedulability in CAN is the exact definition of the set of messages exchanged by the distributed control application. Let $MS = \{m_1, m_2, \ldots, m_n\}$ be this message set, where m_i is the (unique) identifier of the ith message in MS. From the point of view of the schedulability analysis, each message m ($m \in MS$) is completely described by means of the following parameters:

- *Period* (T_m): Cycle time for periodically exchanged messages. For messages generated sporadically, instead, it specifies the minimum interarrival time (i.e., the shortest time between any two consecutive generations of the same message).
- *Deadline* (D_m): Maximum time for completing the reception of message m by the intended destination node(s). If exceeded, the correct behavior of the control application is no longer ensured (and safety problems might occur as well); this is clearly a condition that should never happen in properly operated systems.
- *Jitter* (J_m): This parameter models all latencies that are not directly caused by the communication protocol. It mainly depends on the actual implementation of network controllers and is basically related to hardware overheads. In particular, J_m represents the worst-case delay they may affect message m.
- *Size* (S_m): Specifies the size (in bytes) of the message (or, better, of its payload). In real applications a message may include one or more signals, each one encoded on several bits.

All of the above parameters are defined in the design phase of the control system, and do not depend specifically on the characteristics of the network used to interconnect devices. Starting from them several other parameters can be derived, that will be used directly to assess the network schedulability:

- *Transmission time* (C_m): Time taken to transmit serially over the CAN bus all the bits of the frame that embeds message m (when the transmitting node wins the arbitration)
- *Worst-case response time* (R_m): Maximum end-to-end response time that message m may experience under the given conditions (message set MS and bit rate R on the network)
- *Worst-case queuing delay* (w_m), also known as *transmission delay*: Maximum delay the transmitting node of message m may experience before it wins the arbitration (or, better, before it starts the successful transmission of the frame)
- *Utilization* (U_{MS}): Fraction of the available network bandwidth needed to exchange all the messages included in the message set

The frame transmission time C_m can be computed directly from the message size S_m by taking into account the overhead due to the protocol control information added by CAN. Besides fixed size fields (e.g., identifier, DLC, CRC, etc.), a variable number of stuff bits are inserted into the transmitted frame by the bit stuffing mechanism of CAN, which depends on the actual content of the frame. As we want schedulability to be always ensured, the worst case has to be considered (i.e., when the maximum number of stuff bits are added).

Each CAN frame includes 47 bits of protocol control information (i.e., SOF, arbitration, control, CRC, ACK, and EOF fields, as well as intermission). The bit stuffing mechanism affects 34 of those bits (from the SOF bit up to the CRC field), and the data field (payload) too. Up to one stuff bit could be added every 4 bits of the original frame (an exception is at the very beginning of the frame, where the stuff bit can be added only after 5 bits), and this means that the (worst-case) transmission time C_m is simply given by

$$C_m = \left(34 + 8 \cdot S_m + \left\lfloor \frac{34 + 8 \cdot S_m - 1}{4} \right\rfloor + 13\right) \cdot T_{\text{bit}} = \left(55 + 10 \cdot S_m\right) \cdot T_{\text{bit}}$$

where

R is the transmission speed

$T_{\text{bit}} = 1/R$ is the bit time in the network

A message set is said to be schedulable if the response time of every message in the set is always (i.e., in the worst case, too) lower than or equal to its deadline. This means, that the condition

$$R_m \leq D_m$$

must hold for every message m in MS.

Schedulability in a CAN network is tightly related to the network utilization U_{MS}. This parameter can be easily computed from the transmission time and period of the messages in the message set as

$$U_{MS} = \sum_{m \in MS} \frac{C_m}{T_m}$$

Obviously, utilization must be strictly less than 1 in order to ensure feasibility of the schedule (this means, the offered load is not allowed to exceed the available bandwidth). It is worth noting, however, that despite necessary this condition is not sufficient to ensure schedulability.

Let $t_{\text{req}(m)}$ be the time the transmission request for message m is issued in the originating node, $t_{\text{start}(m)}$ the time the transmission on the bus of the related frame can be first started (irrespective of whether or not the transmitting node will win the contention), $t_{\text{win}(m)}$ the time the successful

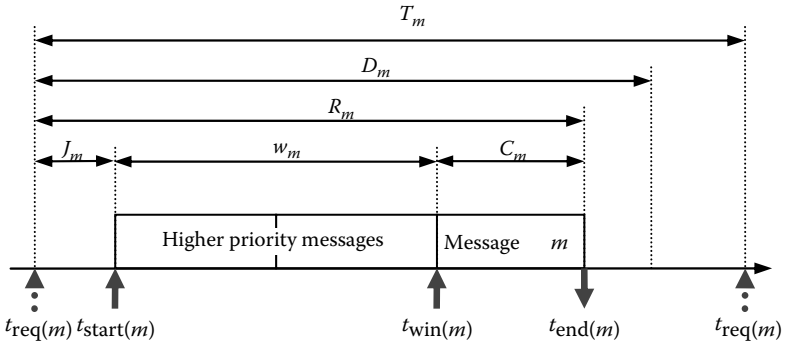


FIGURE 15.6 Worst-case response time in CAN.

transmission on the bus is started (i.e., when the node will effectively win the contention), and $t_{end(m)}$ the time the frame is completely received at the intended destination(s).

As shown in Figure 15.6, the worst-case response time R_m for message m can be evaluated as the sum of the worst-case jitter J_m introduced by the node hardware and software, the delay w_m (possibly) incurred because of lost contentions, and the time C_m taken to transmit serially over the bus all the bits of the frame, that is

$$R_m = J_m + w_m + C_m$$

15.3.2 Schedulability Analysis

As it can be seen, the problem is now evaluating the transmission delay w_m for each message in the set. Delay w_m basically includes two kinds of contributions: at most one message with a priority lower than m, which is already in the process of being transmitted at time $t_{start(m)}$, and every message with a priority higher than m possibly generated in the time interval when message m is queued waiting to be sent. It is worth remembering that in CAN the (numerically) lower the identifier, the higher the priority.

The first kind of contribution gives rise to the so called blocking time (B_m). It can be computed as the maximum transmission time, evaluated among every message k where k is (numerically) greater than m

$$B_m = \max_{k>m} (C_k)$$

The latter contribution, instead, is known as interference. Every message j, whose priority is higher than m, that is generated (or is queued waiting for transmission) in the time interval from $t_{start(m)}$ to one bit after $t_{win(m)}$, in fact, will win the contention with message m and delay it.

As depicted in Figure 15.7, the number $h_{m,j}$ of times the higher-priority message j may interfere with message m basically depends on the duration of this time interval and the generation rate of message j

$$h_{m,j} = \left\lceil \frac{w_m + J_j + t_{bit}}{T_j} \right\rceil$$

In the example shown in Figure 15.7, message j interferes with message m three times (each time making it lose the contention). On the contrary, the fourth instance of message j no longer causes

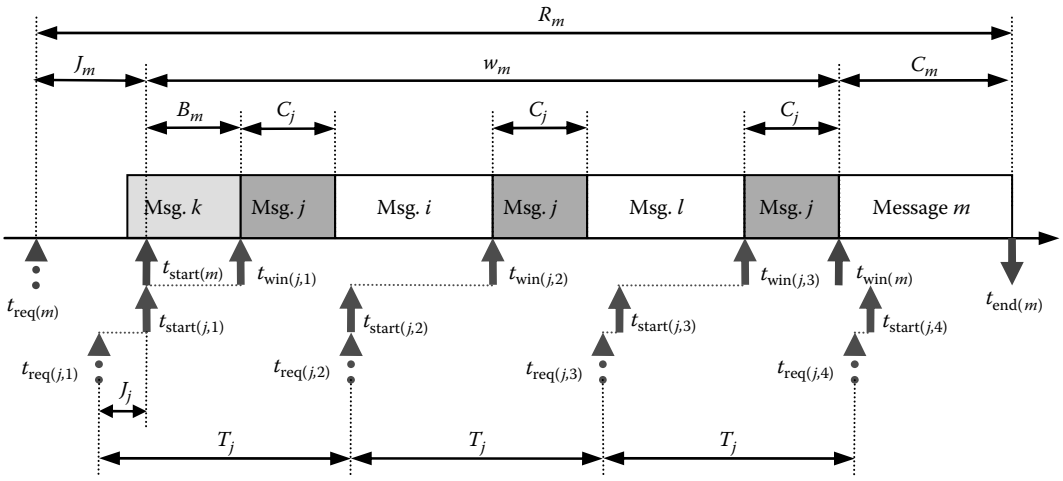


FIGURE 15.7 Blocking time and interference.

interference, as the related transmission request takes place after the SOF bit of message m has been successfully sent (and no other higher priority message is queued waiting for transmission at that time). It is worth noting that the generation of interfering messages could be affected by jitters as well, because of the overhead of network controllers (e.g., J_j in Figure 15.7).

At the same time, however, the queuing delay w_m depends on both the blocking time B_m and the interference caused by all higher priority messages, which in turn depends on h parameters and their duration C

$$w_m = B_m + \sum_{j<m} \left(h_{m,j} \cdot C_j \right)$$

This leads to the following recurrence relation, which provides the transmission delay at step $n+1$ given the one at step n

$$w_m^{n+1} = B_m + \sum_{j<m} \left\lceil \frac{w_m^n + J_j + t_{\text{bit}}}{T_j} \right\rceil \cdot C_j$$

From a practical point of view, the evaluation of w_m proceeds as follow: the initial value w_m^0 is set to 0, then the recurrence relation is repeatedly evaluated until either the value of w_m settles (i.e., w_m^{n+1} equals w_m^n) or the deadline D_m is exceeded. Convergence of this iterative approach is ensured, provided that network utilization U_{MS} is below one.

The above technique enables the evaluation of the transmission delay (and, consequently, of the worst-case transmission time) for every single message. Then, by checking it against message deadlines, the schedulability of the whole message set MS can be assessed.

In 2007, Davis et al. [DAV07] found that, in some cases, the above analysis is optimistic. Despite it was able to correct the flaws in the original method, the newer version of the schedulability test they introduced was more complex, hence it lost a bit in usability. If approximate results are acceptable, an easy way to compute (correctly) the recurrence relation is the following

$$w_m^{n+1} = B_{\text{MAX}} + \sum_{j<m} \left\lceil \frac{w_m^n + J_j + t_{\text{bit}}}{T_j} \right\rceil \cdot C_j$$

where B_{MAX} is the transmission time of the longest message in the considered network and is, in its turn, upper bounded by the duration of the largest CAN frame (8-byte payload). In this case, the obtained results are pessimistic.

15.3.3 Usage

Schedulability analysis has been (and still is) widely used for designing networked embedded system based on CAN, to check whether or not the network is correctly configured and/or dimensioned. It can be applied in all systems where communications are based on asynchronous data exchanges (either cyclic or sporadic). One of the most interesting (and popular) real-life examples of this kind of systems are in-vehicle networks used in the automotive domain. In the case of production cars, in fact, the system is made up of several ECUs (from about 10 up to slightly less than 100), each one of which produces several messages according to either a periodic or a sporadic schedule.

It is worth noting that, from the point of view of communications, data exchanges have been assumed uncorrelated (indeed, they can be correlated at the application level). In this case, the previous analysis is perfectly suitable to assess whether or not timing constraints are met.

Besides schedulability analysis, several concepts introduced above have to be considered carefully in the design phase, particularly when assigning identifiers to different messages that have to be exchanged in a real-time distributed control application. From an intuitive point of view, the most urgent messages (i.e., those characterized by either tightest deadlines or higher generation rates) should be assigned the lowest identifiers (e.g., identifier 0 labels the message which has the highest priority in any CAN network).

If the period of cyclic data exchanges (or the minimum interarrival time of acyclic exchanges) is known in advance, a number of techniques based on either the rate monotonic or deadline monotonic approaches, that have appeared in the literature [TIN94a], can be profitably used to find (if it exists) an optimal assignment of identifiers to messages, so that the resulting schedule is feasible (i.e., deadlines of all the messages are always respected).

15.4 Considerations about CAN

The medium access technique, which CAN relies on, basically implements a nonpreemptive distributed priority-based communication system, where each node is enabled to compete directly for the bus ownership so that it can send messages on its own (this means that CAN is a true multi-master network). This can be advantageous for use in event-driven systems.

15.4.1 Advantages

The CAN protocol is particularly suitable for networked embedded systems, as many (inexpensive) microcontrollers currently exist that include one (or more) CAN controllers, which can be readily embedded both in existing or new projects. This aspect, together with the fact that the protocol is mature, stable and very well-known, makes CAN a viable solution for several years, in spite of newer real-time protocols that feature transmission speeds one-to-two orders of magnitude higher than CAN.

Compared to traditional field networks, CAN is by far simpler and more robust than the token-based access schemes (such as, for example, PROFIBUS when used in multi-master configurations). In fact, there is no need to build or maintain the logical ring, nor to manage the circulation of the token around the master stations. In the same way, it is noticeably more flexible than solutions based on the time division multiple access (TDMA) or combined-message approaches—two techniques adopted by SERCOS and INTERBUS, respectively. This is because message exchanges do not have

to be known in advance. With respect to schemes based on centralized-polling such as FIP, it is not necessary to have a node in the network that acts as the bus arbiter, which can become a single point of failure for the whole system.

All CAN nodes are masters (at least, from the point of view of the MAC mechanism), so it is very simple for them to notify asynchronous events, such as, for example, alarms or critical error conditions. In all cases where this aspect is important, CAN is clearly better than the above-mentioned solutions. Thanks to the arbitration scheme, no message can be delayed by lower priority exchanges (this phenomenon is known as priority inversion). Since communication protocols are not preemptive, a message can still be delayed by a lower priority transmission which has already been started. However, as the frame size in CAN is very small (standard frames are 135-bit long at most, including stuff bits), the blocking time experienced by very urgent messages is quite low (usually, well below 1 ms). This makes CAN a very responsive network, which explains why it is used in many real-time control systems despite its relatively low bandwidth.

When CAN is compared to newer communication solutions, things are a bit different. In spite of much higher bit rates, which make them suitable for the use in industrial plants and factory automation environments, industrial Ethernet solutions are not currently employed (or, better, they are not very popular) in networked embedded systems. This is due to several reasons, including higher costs and the fact that such solutions are, in many cases, not completely settled yet. In the same way, CAN proves to be cheaper and much simpler than solutions purposely designed for the use in automotive systems (e.g., in-vehicle networks), such as, for example, FlexRay. There is no doubt that in the future such technologies will take over CAN in a number of application fields. But, likely, this will not happen tomorrow and, as stated in [FRE02], CAN will remain in use for many years to come.

15.4.2 Drawbacks

There are a number of drawbacks that affect CAN, the most important being related to performance, determinism, and dependability. Though they were initially considered mostly irrelevant, as the time elapsed they have become more and more limiting in a number of application fields, e.g., those related to the advanced in-vehicle x-by-wire systems found in the automotive domain.

15.4.2.1 Performance

Even though inherently efficient and elegant, the arbitration technique of CAN poses serious limitations on the performance that can be obtained by the network. In fact, in order to grant the correct behavior of the arbitration mechanism, the signal must be able to propagate from a node located at one end of the bus up to the farthest node (at the other end) and come back to the first node before this one samples the level on the bus (every node should see the same logical value at any bit time).

Since the sampling point is located (roughly) just before the end of each bit (the exact position can be programmed by means of suitable registers), the end-to-end propagation delay, also including hardware delays of couplers, transceivers, and controllers, must be strictly shorter than about half the bit time (the exact value depending on the bit-timing configuration in the CAN controller and the actual hardware-induced overheads).

As the propagation speed of signals is fixed (about 200 m/μs on copper wires), this implies that the maximum length allowed for the bus is necessarily limited, and depends directly on the bit rate chosen for the network. For example, a 250 kbps CAN network can span at most 200 m; similarly, the maximum bus length allowed when the bit rate is selected as equal to 1 Mbps is less than 40 m. This, to some degree, explains why the maximum bit rate allowed by the CAN specification [ISO1] has been limited to 1 Mbps.

It is worth noting that this limitation depends on physical factors, and hence it cannot be overcome in any way by advances in the technology of transceivers (to make a comparison, it should

be mentioned that, at present, several inexpensive communication technologies are available on the market that allow bit rates in the order of tens or hundreds of Mbps).

Even though this may appear to be a very limiting factor, it will probably not have any relevant impact in the near future for several application areas—including some automotive and process control applications—where a cheap and well-assessed technology is more important than performance. However, there is no doubt that CAN will suffer in a couple of years from the higher bit rates of its competitors, i.e., PROFIBUS-DP (up to 12 Mbps), SERCOS (up to 16 Mbps), FlexRay (up to 10 Mbps), and industrial Ethernet (up to 100 Mbps). Such solutions, in fact, are able to provide a noticeably higher data rate, which is necessary for systems made up of lots of devices or having very short cycle times (1 ms or less).

15.4.2.2 Determinism

Because of its nondestructive bitwise arbitration scheme, CAN is able to resolve in a deterministic way any collision that might occur on the bus. As shown in Section 15.3 about schedulability analysis, this implies that it is possible to design a CAN system so that messages are always delivered before their deadlines (or, equivalently, to ensure that the transmission latencies experienced by every message are upper-bounded). However, if nodes are allowed to produce asynchronous messages on their own—this is the way event-driven systems usually operate—there is no way to know in advance the exact time when a given message will be sent. This is because it is not possible to foresee the actual number of collisions (i.e., interference) a node will experience with higher priority messages. This behavior leads to potentially dangerous jitters, which in some kinds of applications—such as, for example, those involved in the automotive field—might affect the control algorithms in a negative way and worsen their precision. Moreover, if transmission errors due to electromagnetic interferences are taken into account, it might happen that some messages miss their intended deadline.

Another problem related to determinism is that composability is not ensured in CAN networks. This means, that when several subsystems are developed separately and then interconnected to the same network, the overall system may fail to satisfy some timing requirements, even though each subsystem was tested separately and proved to behave correctly and the overall network bandwidth is theoretically sufficient to carry out all data exchanges. This is a severe limitation to the chance of integrating subsystems made from different vendors, and hence makes the design and test tasks much more difficult.

15.4.2.3 Dependability

The last drawback of CAN concerns dependability. Whenever safety-critical applications are considered, where a communication error may lead to damages to the equipment and even injuries to humans (such as, for instance, in most automotive x-by-wire systems), a highly dependable network has to be adopted.

Reliable error detection should be carried out both in the value and in the time domain. In the former case conventional techniques such as, for example, the use of a suitable CRC are adequate. In the latter case, instead, a time-triggered approach is certainly more appropriate than the event-driven communication scheme provided by CAN. In time-triggered systems all actions (including message exchanges, sampling of sensors, actuation of commanded values, and task activations) are known in advance and must take place at precise points in time. In this context, even the presence (or absence) of a message at a given instant provides significant information (i.e., it enables the discovery of faults). For example, masquerading faults (a node that sends messages on behalf of another one) can be avoided in time-triggered networks, and a fail silent behavior can be easily enforced for nodes.

Also the so-called "babbling idiot" problem can affect CAN systems. In fact, a faulty node that repeatedly keeps transmitting a very high priority message on the bus (usually because of a flaw in

the software of the generating task) can slow down and even block the whole network. Such a kind of failures cannot be detected by the fault confinement unit embedded in CAN chips, as they do not depend on physical faults but are due to logical errors. Although bus guardians can be used to overcome these errors, they cannot be implemented efficiently for conventional "event-driven" CAN.

15.4.3 Solutions

The drawbacks highlighted in Section 15.4.2, despite serious, are not going to rule out CAN completely. In many cases, it is (and will remain for several years) the solution of choice to implement cheap and robust networked embedded systems (including those used in automotive applications).

Among the solutions conceived to enhance the behavior of CAN there is the so-called time-triggered CAN (TTCAN) protocol [ISO4], for which network controllers are already available. By adopting a common clock and a time-triggered approach, it achieves very-low jitters and provides a fully deterministic behavior.

Concerning fault tolerance, several interesting proposals have appeared in the literature that can be effectively adopted in real-world systems. For example, in [PRO06] a novel architecture for CAN, which is based on a star topology, is described. Unlike the conventional bus topology, such an arrangement helps avoiding partitioning faults (i.e., a broken link only affects one node and does not prevent the others from communicating).

Finally, it is worth remarking that little can be actually done in order to improve the network performance. Despite some solutions have appeared in the literature—such as, for example, Wide-CAN [CEN02] that provides higher bit rates while still relying on the conventional CAN arbitration technique—their interest is mainly theoretical.

15.5 Time-Triggered CAN

The TTCAN protocol was introduced by Bosch in 1999 with the aim of making CAN suitable for the new needs of the automotive industry, and, in particular, to satisfy (at least in part) the requirements of the upcoming x-by-wire systems. However, TTCAN can be profitably used in all those applications characterized by tight timing requirements that demand for a strictly deterministic behavior. In TTCAN, in fact, it is possible to decide exactly the point in time when messages will be exchanged, irrespective of the network load. Moreover, composability is noticeably improved with respect to CAN, so that it is possible to split a system in the design phase into several subsystems that can be developed and tested separately.

The TTCAN specification is now stable and has been standardized by ISO [ISO4]. The main reason that led to the definition of TTCAN was the need to provide improved determinism in communication while maintaining the highest degree of compatibility with existing CAN devices and development tools. In this way, noticeable savings in investments for the communication technology can be achieved.

15.5.1 Main Features

One of the most appealing features of TTCAN is that it allows both event-driven and time-triggered operations to coexist in the same network at the same time. In order to ease a graceful migration from CAN, TTCAN foresees two levels of implementation that are known as level 1 and 2, respectively. Level 1 implements only basic time-triggered communication over CAN. Level 2, which is a proper extension of level 1, also offers a means for maintaining a global system time across the whole network, irrespective of tolerances and drifts of local oscillators. This enables accurate synchronization up to the application level, and hence true time-triggered operations can take place in the system.

The TTCAN protocol is conceptually placed above the unchanged CAN. It allows time-triggered exchanges to take place in a (quasi-)conventional CAN network. It is worth noting that because TTCAN relies directly on CAN (they adopt the same frame format and the same transmission protocol), it suffers from the same performance limitations as the underlying technology. In particular, is it is not practically feasible to increase the transmission speed above 1 Mbps, and the same drawbacks concerning the limited network extension still hold. However, because of the time-triggered paradigm it relies on, TTCAN is able to ensure strictly deterministic communications, which means that, for example, it is suitable for the first generation of drive-by-wire automotive systems—which are provided with hydraulic/mechanical backups. TTCAN will be likely unsuitable for the next generation steer-by-wire applications. In those cases, in fact, requirements on both fault tolerance and available bandwidth are noticeably more severe.

15.5.2 Protocol Specification

Unlike new-generation automotive protocols (e.g., FlexRay [FR]), which rely on distributed synchronization mechanisms, TTCAN is based on a centralized approach, where a special node called the time master (TM) keeps the whole network synchronized by regularly broadcasting a reference message (RM), usually implemented as a high-priority CAN message. Redundant TMs can be envisaged so as to provide increased reliability. In particular, up to eight potential TMs can be deployed in a TTCAN network. At any time only one node is allowed to be the actual (current) TM, whereas the others behave as backup TMs. The procedure for (re-)electing the current TM operates in a completely dynamic fashion, and ensures that a faulty master is replaced on-the-fly.

Transmission of data is organized as a sequence of basic cycles (BCs). Each BC begins with the RM, which is followed by a fixed number of time windows that are configured off-line and can be of the following three types:

- *Exclusive windows*: Each exclusive window is statically reserved to one predefined message, so that collisions cannot occur; they are used for safety-critical data that have to be sent deterministically and with no jitters.
- *Arbitration windows*: Such a kind of windows are not preallocated to any given message, thus different messages may be potentially competing for transmission; in this case, any possible collision that might occur is solved thanks to the nondestructive CAN arbitration scheme.
- *Free windows*: They are reserved for future expansions of TTCAN systems.

The beginning of each time window corresponds to a time trigger defined in one (or more) producing nodes. So that the boundaries of time windows are never exceeded, TTCAN controllers should disable the automatic retransmission feature of CAN (which would send a message again, whenever the contention is lost or transmission errors are detected) by default. The only exception occurs when there are several adjacent arbitrating windows. In this case, they can be merged together so as to provide a single, larger window, which can accommodate a number of asynchronously generated messages in a more flexible way. The only constraint is that, transmission in such merged arbitrating windows is only allowed if the message can be sent completely in the time that is left before the end of the window.

Despite it seamlessly mixes both synchronous and asynchronous messages (in exclusive and arbitrating windows, respectively), TTCAN is indeed very dependable: in fact, besides the intrinsically deterministic time-driven access to the shared communication support, should there be a temporary lack of synchronization and more than one node tries to transmit in the same exclusive window, the arbitrating scheme of CAN is used to solve the collision.

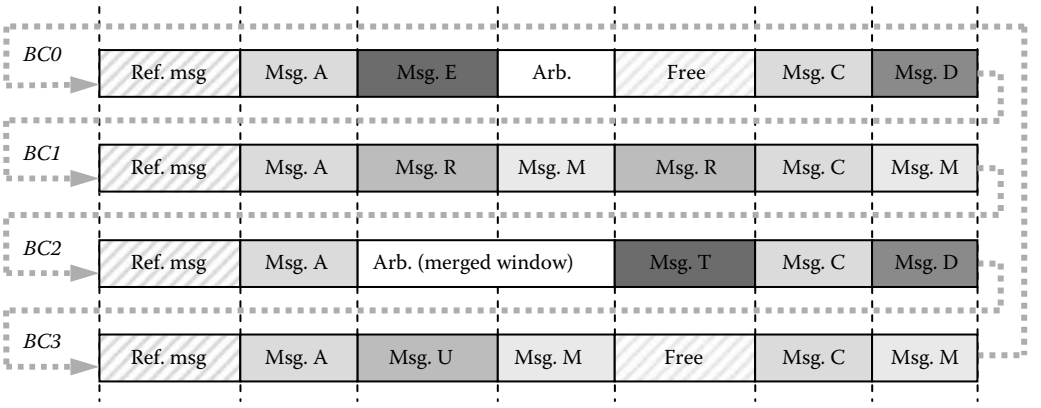


FIGURE 15.8 System matrix in TTCAN.

For increased flexibility, it is possible to have more than one BC. A system matrix can be defined, which consists of up to 64 different BCs (the exact number has to be a power of two), which are repeated periodically (see Figure 15.8). Thus, the actual periodicity in TTCAN is given by the so called matrix cycle. A cycle counter—included in the first byte of RM—is used by every node to determine the current BC. The TM increases the cycle count by one in every cycle, until the maximum value (which is selected on a network-wide basis before operation is started) is reached and the counter is restarted.

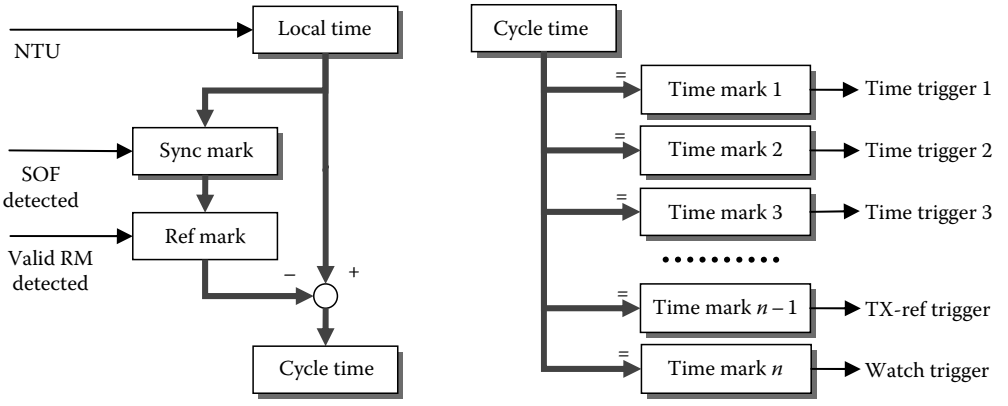
It should be noted that the system matrix is highly column-oriented. In particular, each BC is made up of the same sequence of time windows, i.e., corresponding windows in different BCs do have exactly the same duration. However, they can be used to convey different messages, depending on the cycle counter. In this way, it is possible to have messages in exclusive windows that are repeated once every any given number of BCs. Each one of such messages is assigned a repeat factor and a cycle offset to characterize its transmission schedule. The repeat factor specifies how often a given message has to be sent, whereas the cycle offset is a displacement to avoid collisions among different messages that share the same time window. Both of these parameters are expressed as an integral number of BCs. This "multiplexing" mechanism enables the optimization of the bus allocation with respect to the timing requirements of the different messages, hence achieving noticeable savings for the network bandwidth.

Similarly, more than one exclusive window can be allocated to the same message in the same BC. This is useful either to have redundant transmission of critical data or for refreshing some variables with a rate faster than the BC.

15.5.3 Implementation

Each TTCAN controller has its own clock and is provided with a local time that is basically a counter which is increased by one every network time unit (NTU). The NTU must be the same (at least, nominally) on the whole network. Each node derives it from the local clock and a local parameter known as time unit ratio (TUR). The biggest difference between the level 1 and level 2 variants of the protocol is that the latter defines an adaptive mechanism that, by constantly updating the TUR parameters of the different nodes, is able to keep the NTUs (and, hence, the global time) accurately synchronized.

Whenever receiving RM, each node restarts its cycle timer, so that a common view of the elapsing time is ensured across the whole network. In practice, every time the SOF bit of a new CAN message



(a) Synchronization of the cycle time (b) Scheduling driven by the cycle time

FIGURE 15.9 Time-triggered operation in TTCAN.

is sampled on the bus a synchronization event is generated in every network controller that causes the local time to be copied in a sync mark register. After the message has been completely read, a check is made by controllers: if the SOF bit was related to a valid RM, the sync mark register is loaded into the reference mark register. At this point, the cycle time can be evaluated simply as the difference between the current local time and the reference mark (see Figure 15.9a).

Two kinds of RM are actually foreseen: in level 1 implementations this message is 1-byte long, whereas level 2 relies on a 4-byte RM that is backward compatible with level 1; from a practical point of view, three more bytes are appended for distributing the global time (as seen by the TM).

Protocol execution is driven in every node by the progression of the cycle time. In particular, a number of time marks are defined in each network controller as either transmission or receive triggers, which are used for sending messages and validate message receptions, respectively, as shown in Figure 15.9b. As soon as the cycle time equals a particular time mark, the related trigger is raised. It is worth noting that TTCAN does not require that each node knows the schedule of all messages in the network. Instead, only details of the messages the node sends or is interested in are needed.

Two triggers, that is the TX-ref trigger and the watch trigger, are not used for exchanging data. The former is employed by potential TMs to trigger the generation of RM. The latter, instead, is used to detect the absence of RMs on the network (which implies a faulty condition).

From a general perspective, TTCAN requires slight (and mostly inexpensive) changes to the internal architecture of current CAN chips. In particular, transmit and receive triggers and some counters and registers for managing the cycle time are basically needed for ensuring time-triggered operations.

Even though level 1 could be implemented completely in software, specialized hardware support can reduce noticeably the burden on the processor for managing time-triggered operations. As level 2 compliant controllers have to provide support for drift correction and calibration of the local time, they need the controller hardware to be explicitly modified.

The structure of TTCAN modules is very similar to that of conventional CAN modules. In particular, only two additional blocks are needed, i.e., the trigger memory and the frame synchronization entity. The former is used for storing the time marks of the system matrix that are linked to message buffers held in the controller's memory. The latter, instead, is used to control time-triggered communications.

At present, some controllers are available off-the-shelf that comply with the TTCAN specifications, so that this solution is no longer mainly theoretical, but can be actually embedded in new projects.

15.6 CANopen

In order to reduce the efforts involved in designing and implementing automation and networked embedded systems, a number of higher-level application protocols have been defined in the past few years, which rely on the CAN data-link layer to carry out actual message transfers among the nodes (it is worth remembering that almost all functions of the CAN data-link layer are implemented directly in hardware in the current CAN controllers, and this provides a very good degree of efficiency and reliability for data exchanges). The main aim of such protocols is to provide a usable and well-defined set of service primitives, which can be used to interact with embedded devices in a standardized way. Reduced design and development costs are easily achieved thanks to standardization.

At present, several solutions are available, such as, for instance, CANopen [EN4] for embedded control systems, DeviceNet [EN2] for factory automation, J1939 [J1939] for trucks and other vehicles and so on. Many of them define an abstract model to describe the behavior of devices. This enables interoperability and interchangeability among devices coming from different manufacturers: in fact, as long as a device conforms to a given profile, it can be used in place of any other device (even of a different brand) that complies to the same profile.

In the following, the CANopen protocol will be described, highlighting in particular its use in networked embedded system.

15.6.1 CANopen Basics

CANopen is a CAN-based application layer protocol with highly flexible configuration capabilities that was developed explicitly for embedded systems. It unburdens the developer from dealing with CAN-specific details, such as bit-timing and implementation-specific functions. Moreover, it provides standardized communication objects (COBs) for real-time and configuration data as well as network management (NMT).

CANopen was initially conceived to rely on communication services provided by the CAL [DS20X]. However, current specifications [DS301] no longer refer explicitly to CAL; instead, the relevant services have been included directly in the CANopen documents. CANopen is also known as the international standard EN 50325-4 [EN4]. The set of CANopen specifications includes the application layer, the communication profile as well as several application, device, and interface profiles. These documents are developed and maintained by the CiA member group.

At present, CANopen networks are used in a very broad range of application fields, such as, for example, off-road and rail vehicles, maritime electronics, machine control, medical devices, building automation as well as power generation.

In CANopen, information is exchanged by means of COBs. A number of different COBs are foreseen, which are aimed at different functions:

- *Process data objects* (PDOs), used for real-time exchanges such as, for example, measurements read from sensors and commanded values sent to actuators for controlling the physical system
- *Service data objects* (SDOs), used for nonreal-time communications, e.g., device parameterization and diagnostics
- *Emergency* (EMCY) *objects*, used by devices to notify the occurrence of error conditions
- *Synchronization* (SYNC) *objects*, used to achieve synchronized and coordinate operations in the system, and so on

Even though, in principle, every CAN node is a master (at least from the point of view of the MAC mechanism) CANopen defines a master/slave scheme in order to simplify NMT. In particular, each device is identified by means of a unique 7-bit address, which lies in the range from 1 to 127. So as to

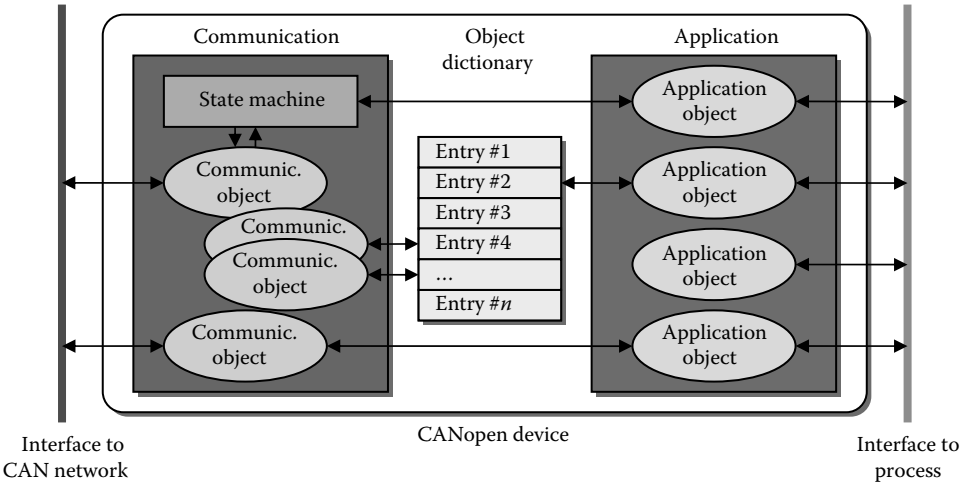


FIGURE 15.10 Conceptual internal architecture of CANopen devices.

ease the system configuration, a predefined master-slave connection set must be provided by every CANopen device, which consists of a standard allocation scheme of CAN identifiers to COBs and is made available directly after initialization.

15.6.2 Object Dictionary

The behavior of every CANopen device is completely described by means of a number of objects, each one tackling a particular aspect related to either the device configuration, the communications on the CAN bus or the functions available to interact with the physical controlled system (e.g., there are objects that define the device type, the manufacturer's name, the hardware and software version, and so on).

All objects relevant to a given node are stored in the object dictionary (OD) of that node. As shown in Figure 15.10, interactions between frames exchanged on the CAN network and the local applications acting on the physical system take place through the OD. This enables a standardized view of devices and their behavior for control applications, and an easy interaction with them through the CAN network.

The OD can be seen as a big table, which stores all information characterizing a device. Both parameterization and process data are kept in the OD, which acts as the sole repository of information for the device.

The OD includes up to 65,536 different objects, and each one of them is addressed through a unique 16 bit address. Any object, in turn, can either be of a simple type (integer, floating point, string, etc.) or consist of up to 256 subentries, each one addressed by means of an 8-bit-long subindex (arrays and records). In the case of simple-type entries, subindex 00_h is used to access the information. This means that each subentry in the OD can be uniquely identified through a 24 bit multiplexer.

From a conceptual point of view, the OD is split into four separate areas, according to the indexes of entries. Entries below 1000_h are used to specify data types. Entries from 1000_h to $1FFF_h$ (communication profile area), instead, are used to describe communication-specific parameters, i.e., they model the interface of the device to the CAN network. These objects are common to all CANopen devices. Entries from 2000_h to $5FFF_h$ (manufacturer-specific profile area) can be freely used by manufacturers to extend the basic set of functions of their devices. Their use has to be considered carefully, in that it could make devices no longer interoperable. Finally, entries from 6000_h to $9FFF_h$ (standardized

profile area) are used to describe all aspects concerning a specific class of devices in a standardized way, as defined in the related device profile.

Other address ranges are either reserved for future use or concern new functionalities that are in the process of being defined at the time of writing.

15.6.3 Process Data Objects

Real-time process data, involved in controlling the physical system, are exchanged in CANopen by means of PDOs according to the producer/consumer scheme. Every PDO is mapped onto exactly one CAN frame, so that it can be exchanged quickly and reliably. As a direct consequence, the amount of data that can be embedded in one PDO is limited to 8 bytes at most. In most cases this is more than sufficient to encode an item of process data.

From the point of view of devices, there are two kinds of PDO: transmit PDO (TPDO) and receive PDO (RPDO). The former kind is produced (i.e., sent) by a node, whereas the latter is consumed by the node itself (this implies they should not be discarded by the frame filtering function).

According to the predefined connection set, each node in CANopen can define up to four RPDOs (from the application master to the device) and four TPDOs (from the device to the application master). In the case more PDOs are needed, the PDO communication parameters of the device (stored as entries in the OD) can be used to define additional messages—or to change the existing ones. This is often sufficient to set up simple master/slave systems in small factory automation systems.

In network embedded systems, which need more flexibility and where device-to-device communication is required as well as multicasting, the PDO linking mechanism can be exploited. From a practical point of view, this consists of configuring explicitly matching PDO communication parameters in two or more devices, so that they are enabled to exchange process data directly according to the standard CAN paradigm.

Finally, by configuring the PDO mapping parameters—if supported by the device—it is possible to define dynamically what application objects (i.e., process variables) will be included in each PDO.

15.6.3.1 PDO Transmission

The transmission of PDOs by CANopen devices can be triggered by some local event taking place on the node—including the expiration of some timeout—or it can be remotely requested from the master. This gives system designers a high degree of flexibility in choosing how devices interact, and enables the features offered by intelligent devices to be exploited in a better way.

No additional control information is added to PDOs by the CANopen protocol, so that the communication efficiency is kept as high as in CAN. The meaning of each PDO is determined directly through the related CAN identifier only. As multicasting is allowed for PDOs, their transmission is unconfirmed, i.e., the producer has no way to know whether or not a PDO was received by all the intended consumers.

One noticeable feature of CANopen is that it can provide synchronous operations as well. In particular, the transmission type of each single PDO can be configured so that exchanges will be driven by the occurrence of a synchronization event on the network, this event being generated regularly by a node known as sync master (usually, it is the same node acting as the application master). Synchronous data exchanges, in this case, take place in periodic communication cycles.

When synchronous operations are selected, commanded values are not actuated by devices immediately as they are received, nor are sampled values transmitted. Instead, as depicted in Figure 15.11, each time a SYNC message is read from the network the synchronous PDOs received in the previous communication cycle are actuated by every output device. At the same time, all sensors sample their input ports and the related values are sent as soon as possible in the next cycle. The synchronous windows length parameter can be defined, which specifies the time when it is certain that all the

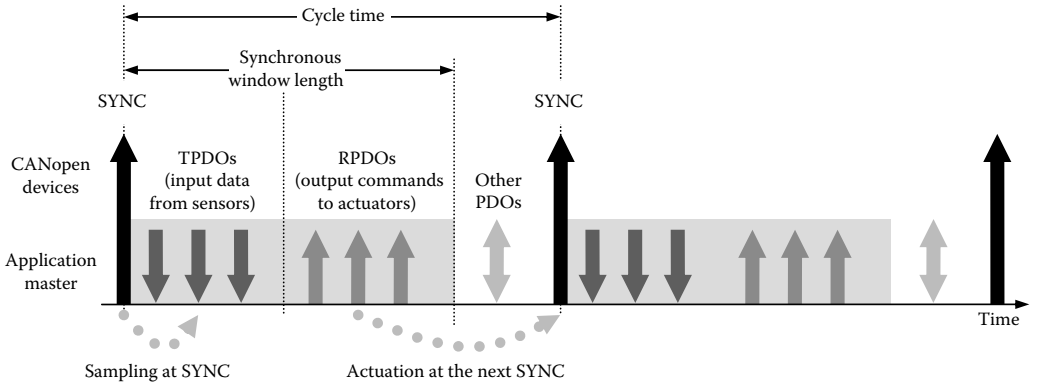


FIGURE 15.11 Synchronous operation.

synchronous values have been exchanged. After this time has elapsed, each device can start processing the input/output values.

Synchronous operations provide a noticeable improvement for lessening the effect of jitters: in this case, in fact, system operations and timings are decoupled from the actual times PDOs are exchanged over the network.

Three triggering modes are defined for PDOs in CANopen that specify conditions for sending a particular PDO over the network (see Figure 15.12):

- *Event- and timer-driven*: In this case, the decision on when a certain data should be sent is completely up to the PDO producer; in particular, the transmission could be triggered either by the occurrence of an application-specific event (depending on the particular device profile or manufacturer) or by the expiration of a local timer because no event has occurred (event time).
- *Remotely requested*: In this case, the transmission of data is triggered by the PDO consumer(s) by means of suitable remote frames (i.e., the PDO is requested explicitly).
- *Synchronous*: In this case, a third entity in the network—other than the PDO producer and its consumers—drives the exchange; in particular, the transmission is triggered by the reception of the SYNC object.

Transmission and triggering modes for TPDOs are configured through the transmission type parameter, encoded as an 8 bit unsigned integer. Values from 0 to 240 mean synchronous transmissions. In particular, value 1 means that the PDO has to be sent at every SYNC, whereas values n in the range 2–240 mean that it is sent every nth SYNC (cyclic transmission). Value 0, instead, is used when generation is acyclic but transmissions have to take place at the SYNC.

Values 252 and 253 denote RTR-only triggering schemes. In particular, value 252 means synchronous RTR-only, that is, data is sampled at the SYNC, temporarily buffered and then sent when the related remote frame is received. Value 253, instead, means asynchronous RTR-only, i.e., both sampling and transmission take place at the remote frame reception.

Finally, values 254 and 255 concern the event-driven triggering schemes. In these cases, triggering is a pure local issue (manufacturer-specific in the former case or device-profile-specific in the latter).

Similar consideration apply to RPDOs, where it is possible to choose between asynchronous and synchronous triggering only. In the asynchronous version the received data is passed to the application (e.g., actuated) as soon as it is received from the network, whereas in the synchronous variant the value is first buffered and then (actually) used at the next SYNC.

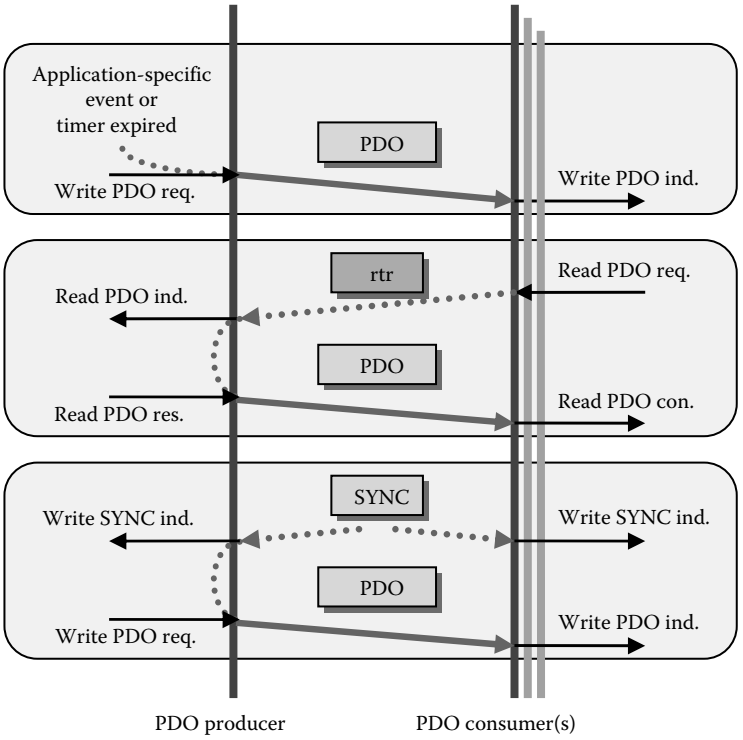


FIGURE 15.12 Transmission modes.

15.6.3.2 PDO Configuration

Details about the way any specific PDO is transmitted over the CAN network are defined through the PDO communication parameter. It includes a lot of information and, in particular:

- CAN-ID used for the CAN frame that embeds the PDO.
- Transmission type, as specified above.
- Inhibit time (expressed as a multiple of 100 μs), which is the shortest time that must elapse between any two transmissions of the PDO. This is useful to prevent high priority messages from hogging all the available bandwidth, thus ruling out lower priority communications.
- Event timer (expressed as a multiple of 1 ms), that is the maximum time between two subsequent transmissions of the same PDO; it can be used to enforce a minimum refresh rate for slowly changing values.
- SYNC start value that is used as a displacement for multiplexing synchronous PDOs whose periodicity is larger than the cycle time.

Communication parameters for RPDOs and TPDOs are stored in the OD in entries 1400_h–$15FF_h$ and 1800_h–$19FF_h$, respectively.

The PDO mapping parameter, instead, specifies how the information stored in the entries of the OD are combined to build the payload of a TPDO (or, equivalently, to decode an RPDO). PDO mapping is useful to improve the protocol efficiency: in fact, it enables several signals in the

same device to be collected in the same CAN frame. Obviously, the information that are gathered in the same PDO must have the same direction (transmission/reception) and triggering scheme (including periodicity). Moreover, their overall size cannot exceed 8 bytes.

Two kinds of PDO mapping can be provided, that is static and dynamic. Static mapping is built-in in the device and cannot be changed, whereas dynamic mapping can be freely configured by accessing the OD through SDOs. Mapping parameters for RPDOs and TPDOs are stored in the OD in entries 1600_h–$17FF_h$ and $1A00_h$–$1BFF_h$, respectively.

15.6.4 Service Data Objects

SDOs provide direct access for reading and writing entries of the OD. They are used in CANopen for parameterization and configuration activities, which usually take place at a lower priority than process data (they are considered nonreal-time exchanges). In this case, a confirmed transmission service is required, which ensures a reliable exchange of information. Furthermore, SDOs are only available on a peer-to-peer client/server communication basis (multicasting is not allowed).

A fragmentation protocol has been defined for SDOs—that comes from the domain transfer services of CAL—so that information of any size can be exchanged. This means that the SDO sender has to split the information is smaller chunks (fragments), which are then reassembled at the receiver's side. This affects the communication efficiency negatively. However, as SDOs are seldom used for the real-time control of the system, this is not perceived as a real problem.

SDOs are used by applications and/or configuration tools to access the entries of the OD. From a practical point of view, two services are provided, which are used to upload and download the content of one subentry of the OD, respectively. SDO transfers are always initiated on the client side of the connection and the related protocol relies on CAN frames having an 8-byte payload.

Three kinds of transfers are envisaged for SDOs, namely expedited, normal (segmented), and block.

An expedited transfer is a particular (simplified) case of SDO that requires only two frame exchanges between the client and the server (i.e., SDO download/upload initiate request and response). As a consequence, however, only up to 4 bytes of data can be exchanged.

A normal transfer is based on a segmented approach. In this case, following the SDO download/upload initiate message pair, a variable number of fragments is sent, each one by means of an SDO download/upload request/response pair.

Block transfer is an optional mode where information is exchanged as a sequence of blocks, each one consisting of a sequence of up to 127 segments. A go-back-n strategy is adopted in this case to confirm the correct delivery of each block, in order to improve the efficiency in transferring large amounts of data.

In CANopen each node must provide SDO server functionalities. This means that it has to define a pair of COB-IDs to support remote access to its OD, one for each direction of the transfer.

Generally speaking, only one SDO client at a time is allowed in the network; in practice, only one SDO client is enabled to connect to any SDO server, and this is implicitly guaranteed with the static approach foreseen in the predefined connection set. Optionally, the dynamic establishment of additional SDO connections can be provided by means of a network entity called the SDO manager.

15.6.5 Other Objects

15.6.5.1 Synchronization Object

The SYNC object is periodically broadcast over the network by the SYNC producer, and is aimed at enabling synchronous operation. The time elapsing between any two subsequent generations of the SYNC message can be configured through the communication cycle period (expressed in μs).

Despite it is assigned a rather high priority, the SYNC object may be nevertheless delayed by other messages (blocking time, because CAN is not preemptive). Hence, the resulting synchronization might not be as accurate as several applications demand (although jitters are much lower than for asynchronous exchanges, they are not usually negligible).

In the initial CANopen specification (as well as in most real devices), the SYNC object was mapped onto an empty data frame: in fact, as it provides a sort of "network clock" to applications, only the time instants when it is received are really meaningful. The most recent specification, however, defines an additional format for the SYNC message that can now include a 1-byte payload. In this case, the data field encodes a counter that is set initially to 1 and then is increased by 1 on every SYNC transmission. When the counter reaches its maximum value (synchronous counter overflow), it is reset back to 1. This permits devices that send synchronous PDOs whose period is greater than the communication cycle to arrange their transmissions (through the SYNC start parameter), so that the overall traffic is evenly spread over different cycles. In turn, this helps reducing jitters and latencies.

15.6.5.2 Emergency Object

The EMCY object is sent asynchronously by devices as a consequence of internal error conditions. It can be considered as a sort of interrupt, which is raised to notify urgent alarms. Its implementation is not mandatory.

The EMCY object is made up of three fields: an error code, encoded on 2 bytes, a 1-byte error register (that specifies errors that are still pending) and a manufacturer specific error field. A number of standardized error code classes are defined in the CANopen communication profile, and others can be added by device profiles.

Only one EMCY object shall be sent for each error event. A state machine is then defined that describes the current error condition of the device. As soon as all errors have been repaired, the device returns to the error-free state.

15.6.6 Network Management

NMT in CANopen relies on a master/slave approach, where one NMT master controls a number of NMT slave devices. Each NMT slave is uniquely identified on the CANopen network by means of a node-ID, encoded as an integer in the range from 1 to 127. NMT requires that one device in the network fulfils the function of the NMT master.

Two kinds of functions concern NMT, namely node control and error control. Node control services, as the name suggests, are used to control the operation of nodes. For example, they can be used to start/stop nodes, to reset them to a predefined state or to put a node in configuration (pre-operational) mode. Error control services, instead, are aimed at verifying if every device is alive and operating properly.

15.6.6.1 Node Control Services

At any time, each CANopen (slave) device can be in one of the following four NMT states, which describe its current behavior and the allowed operations:

- *Initialization*: This state is actually made up of three substates. The first one (initializing), entered autonomously after either power-on or hardware reset, is where basic initialization activities are carried out. Parameters of both the manufacturer-specific and the standardized device-profile areas are set to power-on values in the reset application state, whereas in the reset communication state parameters of the communication profile area are set. If a nonvolatile storage area is provided on the device, such power-on values are

the last stored parameters; on the contrary, default values are used. After initialization is finished, a boot-up message is sent and the preoperational state is entered automatically.

- *Preoperational*: This state is used for configuring the device. As communication through SDOs is available, entries of the OD can be remotely accessed (and changed) by a configuration application/tool. Communication through PDOs, instead, is not allowed yet.
- *Operational*: In this state all communications are permitted, and this usually corresponds to a fully functional behavior for the device. Besides PDO communication, used to exchange set-points and measured values, also SDOs are allowed. However, the access to the OD might be somehow restricted.
- *Stopped*: In this state all communications are completely stopped (except for NMT services). Furthermore, a specific behavior (that usually corresponds to a stopped device) can be enforced, as described by the corresponding device profile.

The resulting behavior of the device can be described by means of the state diagram in Figure 15.13. Transitions are labeled through the following NMT services, whose meaning is self explanatory:

- Start remote node
- Stop remote node
- Enter preoperational
- Reset node
- Reset communication

The NMT master can invoke these services for either one single slave at a time or all of them at the same time (this is useful, e.g., for starting/halting simultaneously all devices in the network). Such

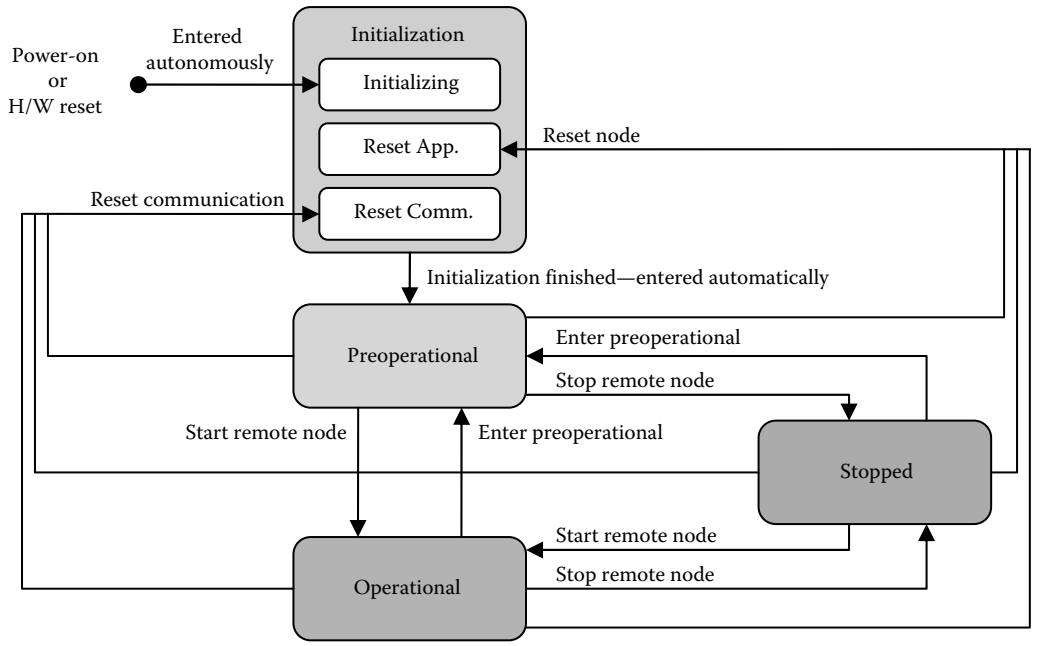


FIGURE 15.13 NMT state machine.

commands are definitely time-critical, and hence they use the highest-priority frames available in CAN (i.e., CAN-ID is set to 0).

The way the NMT state machine is actually coupled to the state machine concerning application task(s) depends on the specific device taken into account, and falls in the scope of device profiles.

15.6.6.2 Error Control Services

Error control services are used to detect possible failures in CANopen, by monitoring the correct operation of the network. Two mechanisms are available, that is node guarding and heartbeat. In both cases, low-priority messages are exchanged periodically in background over the network by the different nodes and suitable watchdogs are defined, both in the NMT master and the NMT slaves. Should one device cease sending these messages, after a given amount of time the NMT layer is made aware of the problem and can take the appropriate actions.

The main difference between the two mechanisms is that in the node guarding protocol the NMT master queries explicitly NMT slaves by means of guarding requests (implemented as remote frames). If a slave does not respond within a predefined time (lifetime) or the state of the slave has changed unexpectedly, the master application is notified. In this case, life guarding can be supported as well (i.e., slave devices guard the correct behavior of the NMT master).

The heartbeat protocol, instead, foresees that heartbeat messages are produced autonomously and periodically by devices, that can be monitored by other nodes in the network. If one of such heartbeat consumers does not hear an heartbeat message from a heartbeat produced within a given time (heartbeat consumer time), the local application is notified.

Messages sent by these two protocols are very similar, and basically encode the NMT state of the transmitting device in 1 byte. The same kind of messages is also used by the boot-up protocol, which is used by slave devices to notify that they have entered the NMT preoperational state after initialization. It is worth noting that the implementation of either guarding or heartbeat is mandatory.

15.6.6.3 Predefined Connection Set

In order to reduce the efforts involved in setting up a CANopen system, a default allocation scheme is provided that maps COBs of the different devices directly onto CAN identifiers. Such an allocation is made available just after initialization (provided that no modifications have been stored in the nonvolatile memory of the device). In many cases, such a scheme is adequate, e.g., when simple applications are taken into account, that consist of a single application master controlling several slave devices.

According to the predefined allocation scheme, the CAN-ID is made up of two subfields: the function code (FC), which takes the four most significant bits of the identifier, and the node-ID that is encoded on the seven least significant bits, as shown in Figure 15.14. The FC basically encodes the type of COB (PDO, SDO, EMCY, NMT, etc.), and most affects the resulting priority of the frame. The node-ID, instead, besides providing a node addressing scheme, ensures that no two devices in the network can send a frame with the same identifier (which could lead to unsolvable collisions on the bus).

Some of the COBs in the predefined connection set, such as NMT, SYNC, and TIME are meant to be sent to all devices (broadcast). The related CAN-IDs can be easily singled out, as the node-ID field is set to 0. In general, it is worth noting that the particular format chosen for identifiers makes the configuration of the filtering masks on CAN controllers easier.

Concerning frames for peer-to peer communications, the scheme basically provides each slave device with (up to) four TPDOs and (up to) four RPDOs for sending/receiving real-time process data, one EMCY object for asynchronous alarm notification, one SDO to access the OD (in the form of a

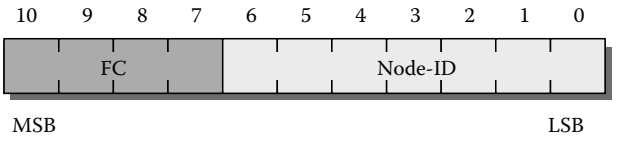


FIGURE 15.14 CAN-ID in the predefined connection set.

TABLE 15.1 COBs Foreseen by the Predefined Connection Set

COB	FC	Addressing	CAN-ID(s)
NMT	0000_b	Broadcast	000_h
SYNC	0001_b	Broadcast	080_h
EMCY	0001_b	Peer-to-peer	081_h–$0FF_h$
TIME	0010_b	Broadcast	100_h
TPDO1	0011_b	Peer-to-peer	181_h–$1FF_h$
RPDO1	0100_b	Peer-to-peer	201_h–$27F_h$
TPDO2	0101_b	Peer-to-peer	281_h–$2FF_h$
RPDO2	0110_b	Peer-to-peer	301_h–$37F_h$
TPDO3	0111_b	Peer-to-peer	381_h–$3FF_h$
RPDO3	1000_b	Peer-to-peer	401_h–$47F_h$
TPDO4	1001_b	Peer-to-peer	481_h–$4FF_h$
RPDO4	1010_b	Peer-to-peer	501_h–$57F_h$
SDO (tx)	1011_b	Peer-to-peer	581_h–$5FF_h$
SDO (rx)	1100_b	Peer-to-peer	601_h–$67F_h$
NMT error control	1110_b	Peer-to-peer	701_h–$77F_h$

pair of CAN messages, so as to provide a bidirectional communication channel), and one low-priority message to be used by either the heartbeat or the node guarding error control mechanisms. The predefined connection set is shown in Table 15.1.

The CAN-ID assigned to the SYNC, TIME, and EMCY objects, as well as to any PDO, can be changed through suitable reconfiguration—if this kind of parameterization is supported by the device.

15.7 CANopen Device Profile for Generic I/O Modules

Besides a common and agreed protocol for exchanging data, a number of device profiles have been standardized in CANopen to ensure interoperability between equipment from different manufacturers. Device profiles are meant to ease the tasks of system integrators when designing and deploying CANopen systems, by providing off-the-shelf devices with true plug & play capabilities.

Each profile describes the common behavior of a particular class of devices, and is usually described in a separate document. Currently, some tens profiles have already been defined, including the following:

- DS 401: I/O devices, including both digital and analog input/output devices
- DS 402: drives and motion control, which is used to describe products for digital motion, such as, for example, frequency inverters, stepper motor controllers, and servo controllers
- DS 404: measuring devices and closed-loop controllers, to measure and control physical quantities
- DS 406: encoders, which defines the behavior of incremental/absolute linear and rotary encoders for measuring position, velocity, and so on

Profiles listed above have been introduced mainly for general-purpose applications. This means, they were not designed to fulfill the requirements of any specific application domain and, as a consequence,

compliant devices are usually used for assembling automated production lines in factory automation systems.

Other profiles have been defined as well, that are aimed at fulfilling specific requirements concerning either the functions or the installation environment of a particular application domain. This is the case, for example, of the CiA 447 application profile defined for the automotive domain that specifies interfaces for add-on devices to be used in special-purpose cars. Devices for police cars (e.g., in the roof bar and for digital radio), for taxi/cabs (e.g., taximeter) and for disabled drivers are included in this profile.

In the following, the CANopen device-profile CiA 401 for generic I/O modules will be described in detail, to provide an insight of what a device profile in CANopen actually is and how it can help designers to build up networked embedded systems.

15.7.1 Device Definition

The device profile for generic I/O modules [DS401] is certainly one of the most popular device profiles defined in CANopen. Generic input/output modules provide digital/analog inputs and/or outputs; this means, they are not bound to application-specific functions, such as, for example, measuring the engine temperature or switching the turn lights on/off.

Many optional configuration parameters are specified by this profile, as well as a number of PDOs to be either sent or received. For instance, it is possible to define the polarity of each digital input/output port or to apply a filtering mask to selected channels. For analog devices, either raw or converted (after a scaling factor and an offset have been applied) values can be used. In the same way, triggering conditions can be defined when specific thresholds are exceeded. The generic I/O modules profile supports different access granularities for digital I/Os and several resolutions (and formats) for analog I/Os.

Input values (from sensors) are sent on the CAN network as TPDOs, whose transmission defaults to the asynchronous (i.e., event-driven) triggering. However, also the synchronous or remotely requested triggering schemes can be optionally enabled, if supported by the device. In the same way, output values are provided (to actuators) via asynchronous RPDOs.

I/O values may be accessed by remote applications through SDOs as well, even though this is not recommended as it leads to a reduced efficiency and generally higher delays.

15.7.1.1 Profile Predefinitions

Several entries of the OD are standardized by this profile. One of the most significant objects in the communication profile area is, perhaps, object 1000_h that defines both the device type and its functionality, each one of them encoded in 2 bytes. In particular, the type is specified through the device-profile number (in this case set to the value 401), whereas the functionality field tells what kind of channels (e.g., input vs. output, digital vs. analog) are effectively included in the device.

Moreover, I/O modules, which comply with this profile, define (part of) the entries of the OD in the standardized profile area in the range from 6000_h to $67FF_h$. For digital modules, different objects are foreseen, which provide access to (exactly the same) process data stored in the OD on either 1, 8, 16, or 32 bits. However, only the 8-bit access scheme is mandatory, whereas the others are optional. Objects 6000_h and 6200_h are used for 8-bit access to digital input and output modules, respectively. Each subentry encodes eight adjacent digital channels.

Concerning analog modules, values encoded as integers on 8, 16, and 32 bits are available. In addition, floating-point and manufacturer-specific formats can be used. Only 16 bit data are mandatory, whereas the others are optional. Objects 6401_h and 6411_h are used for accessing 16 bits analog input and output values, respectively. Each subentry encodes one analog channel.

In the same way, a number of default PDOs are defined, together with a default mapping:

- *RPDO1* (*digital outputs*): Up to 64 digital outputs can be received from the network through this RPDO that are actuated immediately.
- *TPDO1* (*digital inputs*): Up to 64 digital inputs can be sent over the network through this TPDO, according to an event-drive scheme. This means, the PDO is transmitted as soon as one digital input changes its value (provided it is not masked off via the interrupt mask).
- *RPDO2* (*analogue outputs*): Up to 4 analogue outputs—each one encoded on 16 bits—can be received from the network through this RPDO.
- *TPDO2* (*analogue inputs*): Up to 4 analogue inputs—each one encoded on 16 bits—can be sent on the network through this TPDO, according to an event-drive scheme.

Only PDOs concerning the supported functionalities have to be provided effectively. However, additional manufacturer-specific PDOs could be included in the module.

15.7.2 Device Behavior

In the profile specification, also the behavior of devices is defined in a formal way. In the case of generic I/O modules, block diagrams are provided that describe how information is exchanged between the CANopen network and the real world and the way such operations can be configured.

15.7.2.1 Digital Input Devices

In Figure 15.15 the sequence of steps involved in the processing chain of a digital input is shows. Details on the way each operation is carried out can be changed by configuring the appropriate entries of the OD. A filter constant can be activated separately (2) on each single digital input channel (1) by means of object 6003_h; moreover, its polarity can be optionally changed (3) through object 6002_h. Then, all bits associated to the different input channels are stored in the suitable entries of the OD (object 6000_h), from where they will be subsequently extracted and sent over the network (4). In this case, values are first mapped onto the payload of a PDO through the PDO mapping mechanism (5); then the correct identifier is selected (6) by means of the PDO communication parameter.

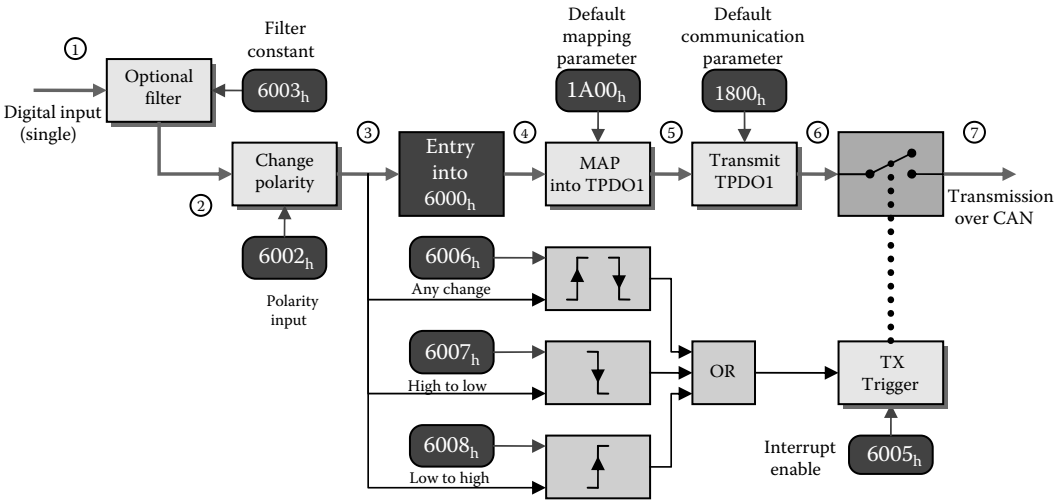


FIGURE 15.15 Conceptual architecture of digital inputs.

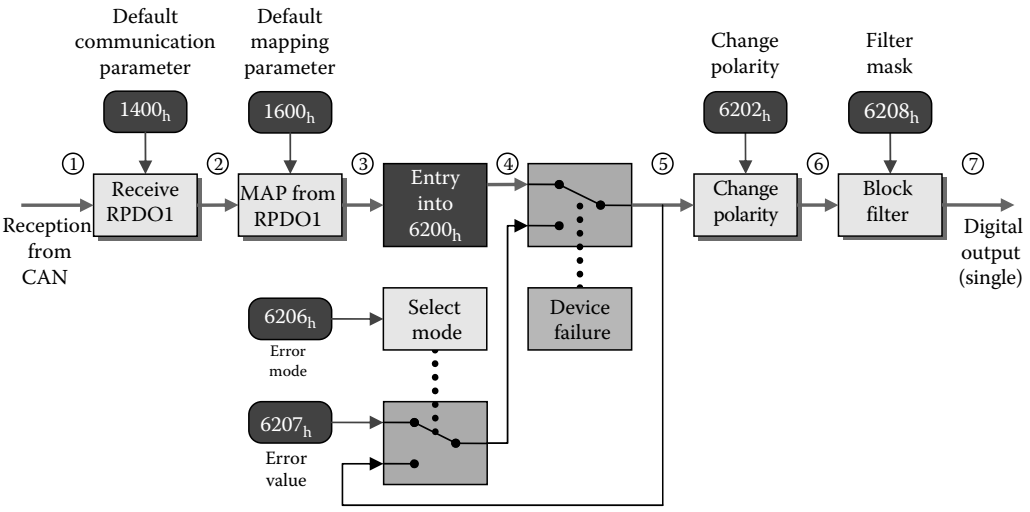


FIGURE 15.16 Conceptual architecture of digital outputs.

The actual transmission on the CAN bus (7) is driven by the trigger block. By default, the asynchronous mode is enabled, that triggers the transmission as soon as any of the input channels changes its logical value. However, this behavior can be changed through objects 6006_h, 6007_h, and 6008_h. Moreover, triggering can be globally enabled/disabled through the (Boolean) object 6005_h.

It is clear that, by writing into the relevant entries of the OD, a control application or a configuration tool can easily customize the behavior of each single digital input channel, as well as decide how these information are sent over the CAN network.

15.7.2.2 Digital Output Devices

Figure 15.16 shows the steps involved in the processing chain affecting the operation of a digital output device. Frames received from the CAN bus (1) are first filtered according to the information stored in the RPDO communication parameter (2). Whenever the received frame corresponds to the RPDO1 configured for the device, its payload is decoded according to the PDO mapping parameters, and the related values are stored (3) in the suitable entry of the OD (object 6200_h). After an optional polarity change (5), configured through the object 6202_h, the output value is made available on the related output channel (7). A filter block is also provided (6) through object 6208_h: when disabled, the previous value is retained on the output channel.

The lower part of Figure 15.16 describes the behavior of the actuator in the case a device internal failure (including heartbeat or life guarding events) is detected. In order to improve safety, two objects (6206_h and 6207_h) are provided that specify the value each digital output channel should assume in the case of device failure. In particular, each output can be set separately to 0, 1, or its own value before the failure was detected. Whenever a failure is detected, the output is automatically switched (4) to such a value.

References

[PRO06] Barranco M., Proenza J., Rodriguez-Navas G., and Almeida L., An active star topology for improving fault confinement in CAN networks, *IEEE Transactions on Industrial Informatics*, 2(2), 78–85, May 2006.

[DS102] CAN in Automation International Users and Manufacturers Group e.V., CAN physical layer for industrial applications—Two-wire differential transmission, CiA DS 102, Version 2.0, April 1994.

[DS20X] CAN in Automation International Users and Manufacturers Group e.V., CAN application layer for industrial applications, CiA DS 201/202/203/204/205/206/207, Version 1.1, February 1996.

[DS301] CAN in Automation International Users and Manufacturers Group e.V., CANopen—Application layer and communication profile, CiA DS 301, Version 4.02, 2002.

[DS401] CAN in Automation International Users and Manufacturers Group e.V., CANopen—Device profile for generic I/O modules, CiA DS 401, Version 2.1, May 2002.

[CEN02] Cena G. and Valenzano A., A multistage hierarchical distributed arbitration technique for priority-based real-time communication systems, *IEEE Transactions on Industrial Electronics*, 49(6), 1227–1239, December 2002, The Institute of Electrical and Electronics Engineers, New York.

[DAV07] Davis R. I., Burns A., Bril R. J., and Lukkien J. J., Controller area network (CAN) schedulability analysis: Refuted, revisited and revised, *Real-Time Systems*, 35(3), 239–272, April 2007, Springer, the Netherlands.

[EN1] European Committee for Electrotechnical Standardisation, Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces—Part 1: General requirements, EN 50325-1:2002, 2002-12-12.

[EN2] European Committee for Electrotechnical Standardisation, Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces—Part 2: DeviceNet, EN 50325-2:2000, 2000-10-30.

[EN3] European Committee for Electrotechnical Standardisation, Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces—Part 3: Smart distributed system (SDS), EN 50325-3:2001, 2001-04-10.

[EN4] European Committee for Electrotechnical Standardisation, Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces—Part 4: CANopen, EN 50325-4:2002, 2002-12-12.

[FR] FlexRay consortium, FlexRay communications system—Protocol specification, Version 2.1, Rev. A, December 2005.

[FRE02] Fredriksson L., CAN for critical embedded automotive networks, *IEEE Micro*, 22(4), 28–35, July–August 2002.

[ISO1] International Standard Organisation, Road vehicles—Controller area network—Part 1: Data link layer and physical signalling, ISO 11898-1:2003, 2003-11-19.

[ISO2] International Standard Organisation, Road vehicles—Controller area network—Part 2: High-speed medium access unit, ISO 11898-2:2003, 2003-11-19.

[ISO3] International Standard Organisation, Road vehicles—Controller area network—Part 3: Low-speed, fault-tolerant, medium dependent interface, ISO 11898-3:2006, 2006-05-24.

[ISO4] International Standard Organisation, Road vehicles—Controller area network—Part 4: Time-triggered communication, ISO 11898-4:2004, 2004-08-05.

[ISO5] International Standard Organisation, Road vehicles—Controller area network—Part 5: High-speed medium access unit with low-power mode, ISO 11898-5:2007, 2007-06-12.

[J1939] SAE International, Core J1939 standards, J1939/-01/11/13/15/21/31/71/73/74/75/81.

[TIN94a] Tindell K. W., and Burns A., Guaranteeing message latencies on controller area network (CAN), in *Proc. of the 1st International CAN Conference*, 1994, pp. 1–11.

[TIN94b] Tindell K. W., Hansson H., and Wellings A. J., Analysing real-time communications: Controller area network (CAN), in *Proc. of the 15th Real-Time Systems Symposium (RTSS'94)*, IEEE Computer Society Press, 1994, pp. 259–263.

[TIN95] Tindell K. W., Burns A., and Wellings A. J., Calculating controller area network (CAN) messages response times, *Control Engineering Practice*, 3(8), 1995, 1163–1169, Elsevier.

16

FlexRay Communication Technology

16.1 Introduction **16**-1
16.2 Automotive Requirements **16**-1
Cutting Costs
16.3 What is FlexRay? **16**-3
Media Access • Clock Synchronization • Start-Up • Coding and Physical Layer • Bus Guardian • Protocol Services • FlexRay Current State
16.4 System Configuration **16**-11
Development Models
16.5 Standard Software Components **16**-15
Standardized Interfaces
References **16**-16

Roman Nossal-Tueyeni
Austro Control GmbH

Dietmar Millinger
Elektrobit Austria GmbH

16.1 Introduction

New electronic technologies have dramatically changed cars and the way we experience driving. ABS, ESP, air bags, and many more applications have made cars a lot more convenient, comfortable, and, above all, safer. This trend of the past decade has been rather pleasant for the consumer, and a tedious task for the automotive industry. The reasons for this drawback have not only to do with the need of higher integration of the involved technologies. The very nature of the deployed communication technologies makes the task of integration itself a lot more complex as well as the design of fault-tolerant systems on top of these communication technologies rather difficult.

These limitations on the one hand as well as requirements and anticipated challenges of future automotive applications on the other hand motivated OEMs and suppliers to join forces. The goal of the 2001 founded FlexRay Consortium [1]: to establish one standard for a high-performance communication technology in the automotive industry.

16.2 Automotive Requirements

Since OEMs and suppliers were the founding fathers of the FlexRay Consortium, it was clear from the very beginning of the work on the new de facto communication standard that FlexRay has to meet the requirements of the automotive industry. Therefore, two key issues have driven the development work for the communication protocol—the need for a technological basis and solution for future safety-related applications and the need to keep costs down.

16.2.1 Cutting Costs

The cost factor is a key driver for many requirements for the communication system as the push for systematic reuse of existing components in multiple car platforms proves. Due to this approach, subsets of components related to a specific function can be reused in multiple platforms without changes inside the components. This elegant and cost-saving solution, however, is only possible if the communication system offers two decisive qualities:

1. It must be standardized and provide a stable interface to the components.
2. Communication system has to provide a deterministic communication service to the components.

This communication determinism is the solution for the problem of interdependencies between components, which is a major issue and cost factor in today's automotive distributed systems. Since any change in one component can change the behavior of the entire system, integration and testing is of utmost importance in order to ensure the needed system reliability. At the same time, testing is extremely difficult and expensive. A deterministic communication system significantly reduces this integration and test effort as it guarantees that the cross-influence is completely under control of the application and not introduced by the communication system. This property is often referred to as "composability," meaning that each component of a system can be tested in isolation and integration of these components does not have any side effects.

16.2.1.1 Migration

A new technology, such as FlexRay, does not make all predecessors obsolete at once. It rather replaces the traditional systems gradually and builds on proven solutions. Therefore, existing components and applications have to be migrated into new systems. In order to create this migration path as smooth and efficient as possible, FlexRay has integrated some key qualities of existing communication technologies, e.g., dynamic communication.

16.2.1.2 Scalability

Communication determinism and reuse are also key enablers for scalability, which obviously is yet another cost-driven requirement. Scalability, however, does not only call for communication determinism and reuse, but also for the support of multiple network topologies and network architectures as well as the applicability of the communication technology in different application domains like power train, chassis control, backbone architectures, or driver assistance systems.

16.2.1.3 Future Proof

Keeping costs down is only one side of the coin. The automotive industry has visions of the future car and applications. The most obvious developments are active safety functions like electronic braking systems, driver convenience functions like active rear steering for parking, or the fast growing domain of driver assistance systems like active cruise control or the lane departure warning function. These automotive applications demand a high level of reliability and safety from the network infrastructure in the car in order to provide the required level of safety at the system level. Therefore, the communication technology has to meet requirements such as redundant communication channels, deterministic media access scheme, high robustness in case of transient faults, distributed fault-tolerant agreement about the protocol state, and extensive error detection and reporting toward the application. The most stringent particular requirement arises from the deterministic media access scheme. In time division multiple access (TDMA) schemes for networks, all participating communication partners need a common understanding of the time used in order to control the access to the communication medium. Typically a fault-tolerant distributed mechanism for clock synchronization

is required. Additionally, the safety requirement introduces the need to protect individual communication partners from faults of other partners by means of so-called guardians. Otherwise, errors of one partner could cross-influence other partners thus violating safety demands.

Specific automotive issues complete the broad range of requirements forming the framework for and of FlexRay. These issues range from the use of automotive components like X-tals, automotive electro-magnetic compatibility (EMC) requirements, support for power management to conserve battery power, support for electrical and optical physical layers, to a high bandwidth demand of at least two times 10 Mbps.

16.3 What is FlexRay?

Before the development of FlexRay was started, a comprehensive evaluation of the existing technologies took place. The results showed that none of the existing communication technologies could fulfill the requirements to a satisfactory degree. Thus the development of a new technology was started. The resulting communication protocol FlexRay is an open, scalable, deterministic, and high-performance communication technology for automotive applications.

A FlexRay network consists of a set of electronic control units (ECUs) with integrated communication controllers (CCs) (Figure 16.1). Each CC connects the ECU to one or more communication channels via a communication port, which in turn links to a bus driver (BD). The BD connects to the physical layer of the communication channel and can contain a guardian unit that monitors the TDMA access of the controller (the architecture of an ECU is depicted in Figure 16.2). A communication channel can be as simple as a single bus wire but also be as complex as active or passive star configurations.

FlexRay supports the operation of a CC with single or redundant communication channels. In case of single communication channel configuration, all controllers are attached to the communication channel via one port. In case of redundant configuration, controllers can be attached to the communication channels via one or two ports. Controllers that are connected to two channels can be configured to transmit data redundantly on two channels at the same time. This redundant transmission allows the masking of a temporary fault of one communication channel and thus constitutes a powerful fault-tolerance feature of the protocol. A second fault-tolerance feature related to transient faults can be constructed by the redundant transmission of data over the same channels with a particular time delay between the redundant transmissions. This delayed transmission allows tolerating transient faults on both channels under particular preconditions.

If the fault-tolerance property of two independent channels is not required for a specific application, the channels can be used to transfer different data, thus effectively doubling the transmission bandwidth.

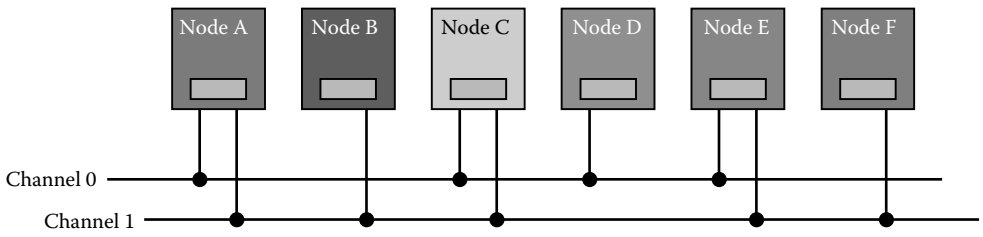


FIGURE 16.1 FlexRay network.

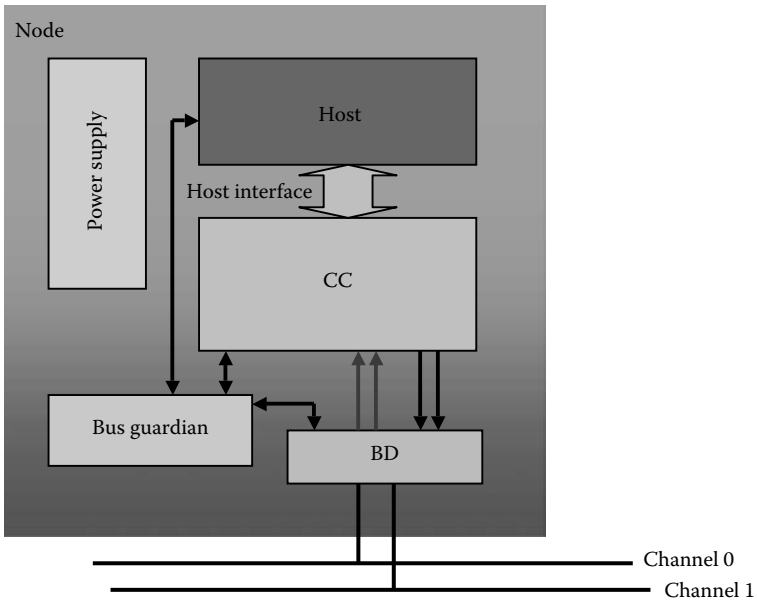


FIGURE 16.2 ECU architecture.

16.3.1 Media Access

The media access strategy of FlexRay is basically a TDMA scheme with some very specific properties. The basic element of the TDMA scheme is a communication cycle. A communication cycle contains a static segment, a dynamic segment, and two protocol segments called symbol window and network idle time (Figure 16.3). Communication cycles are executed periodically from start-up of the network until shutdown. Two or more communication cycles can form an application cycle.

The static segment consists of slots with fixed duration. The duration and the number of slots are determined by configuration parameters of the FlexRay controllers. These parameters must be identical in all controllers of a network. They form a so-called global contract. Each slot is exclusively owned

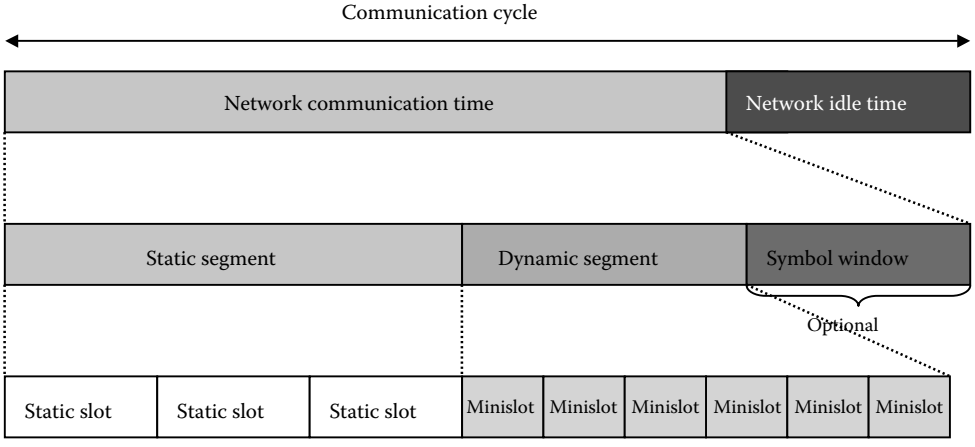


FIGURE 16.3 FlexRay communication cycle.

by one FlexRay CC for transmission of a frame. This ownership only relates to one channel. On other channels, in the same slot either the same or another controller can transmit a frame. The identification of the transmitting controllers in one slot is also determined by configuration parameters of the FlexRay controllers. This piece of information is local to the sending controller. The receiving controllers do not possess any knowledge on the transmitter of a frame nor on its content; they are configured solely to receive in a specific slot. Hence the content of a frame is determined by its positions in the communication cycle.

Alternatively, a message ID in the payload section of the frame can be used to identify a message. The message ID uniquely tells the receivers which information is contained in a frame. By providing a filter mechanism for message IDs, a controller can pick specific data.

The static segment provides deterministic communication timing, since it is exactly known when a frame is transmitted on the channel, giving a strong guarantee for the communication latency. This strong guarantee in the static segment comes for the trade-off of fixed bandwidth reservation.

The dynamic segment has fixed duration, which is subdivided into so-called minislots. A minislot has a fixed length that is substantially shorter than that of a static slot. The length of a minislot is not sufficient to accommodate a frame; a minislot only defines a potential start time of a transmission in the dynamic segment. Similar to static slots, each minislot is exclusively owned by one FlexRay controller for the transmission of a frame. During the dynamic segment, all controllers in the network maintain a consistent view about the current minislot. If a controller wants to transmit in a minislot, the controller accesses the medium and starts transmitting the frame. This is detected by all other controllers, which interrupt the counting of minislots. Thus, the minislot is "expanded" to a real slot, which is large enough to accommodate a frame transmission. It is only after the end of the frame transmission that counting of the minislots continues. The expansion of a minislot reduced the number of minislots available in this dynamic segment. The operation of the dynamic segment is illustrated in Figure 16.4: Figure 16.4a shows the situation before minislot 4 occurs. Each of the channels offers 16 minislots for transmission. The owner of minislot 4 on channel 0—in this case controller D—has data to transmit. Hence the minislot is expanded as shown in Figure 16.4b. The number of available minislots on channel 0 is reduced to 13.

If there are no data to transmit by the owner of a minislot, it remains silent. The minislot is not expanded and slot counting continues with the next minislot. As no minislot expansion occurred, no additional bandwidth beyond the minislot itself is used; hence other lower-priority minislots that are sequenced later within the dynamic segment have more bandwidth available.

This dynamic media access control scheme produces a priority and demand-driven access pattern that optimally uses the reserved bandwidth for dynamic communication. A controller that owns an "earlier" minislot, i.e., a minislot, which has a lower number, has higher priority. The further in the dynamic segment a minislot is situated, the higher is the probability that it will not be in existence in a particular cycle due to the expansion of higher priority slots. A minislot is only expanded and its bandwidth used if the owning controller has data to transmit. As a consequence, the local controller configuration has to ensure that each minislot is configured only once in a network. The minimum duration of a minislot is mainly determined by physical parameters of the network (delay) and by the maximum deviation of the clock frequency in the controllers. The duration of a minislot and the length of the dynamic segment are global configuration parameters that have to be consistent within all controllers in the network.

The symbol window is a time slot of fixed duration, in which special symbols can be transmitted on the network. Symbols are used for network management purposes.

The network idle time is a protocol-specific time window, in which no traffic is scheduled on the communication channel. The CCs use this time window to execute the clock synchronization algorithm. The offset correction (see below) that is done as a consequence of clock synchronization requires that some controllers correct their local view of the time forward and others have to correct backward. The correction is done in the network idle time. Hence no consistent operations of

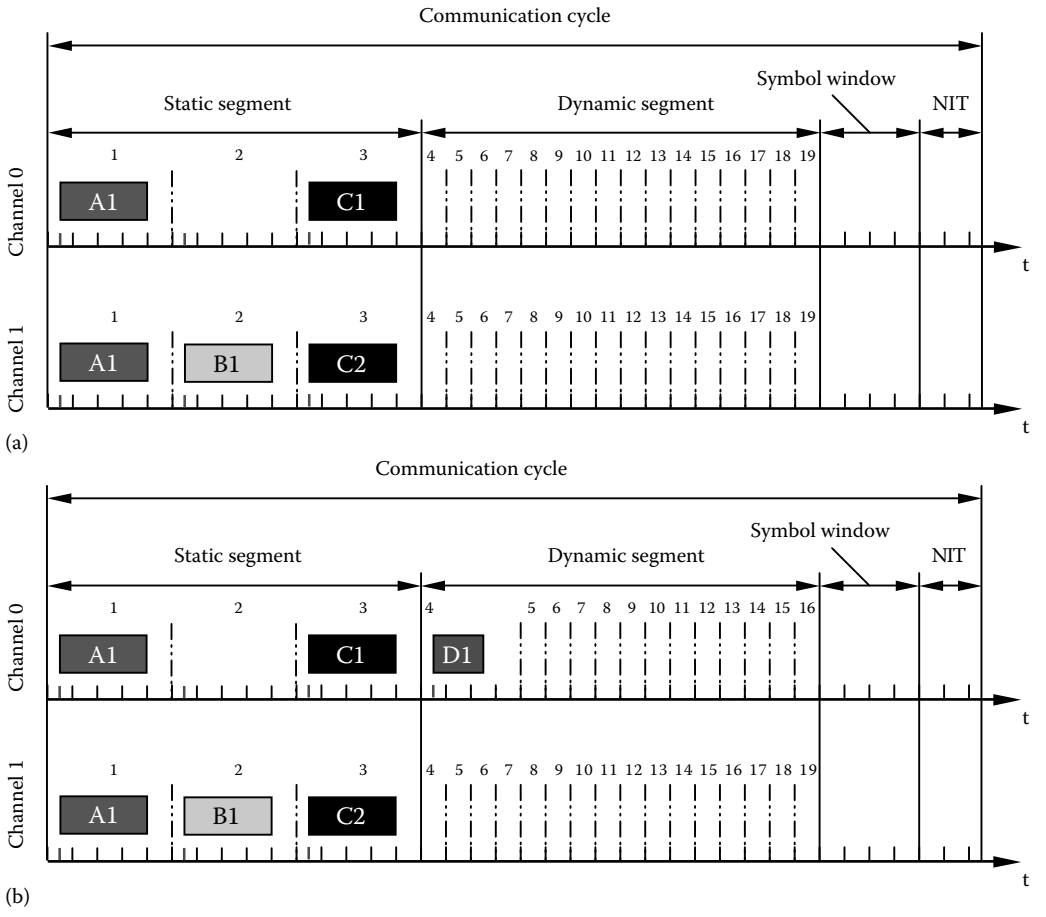


FIGURE 16.4 FlexRay dynamic segment.

the media access control can be guaranteed and thus silence is required. Since this duration has to be subtracted from net bandwidth, this duration is kept as small as possible. The minimum length is largely determined by the maximum deviation between the local clocks after one communication cycle. The duration of the network idle time is a global parameter that has to be consistent between all controllers in a network.

While at least a minimal static segment and the network idle time are mandatory parts of a communication cycle, the symbol window and the dynamic segment are optional parts. This result in basically three reasonable configurations (see Figure 16.5):

- Pure static configuration, which contains only static slots for transmission. In order to enable clock synchronization, the static segment must consist of at least two slots, which are owned by different controllers. If a fault-tolerant clock synchronization should be maintained, the static segment must comprise at least four static slots.
- Mixed configuration with a static segment and a dynamic segment, where the ratio between static bandwidth and dynamic bandwidth can vary in a broad range.
- Finally, a pure dynamic configuration with all bandwidth assigned to dynamic communication. This configuration also requires a so-called degraded static segment, which has two static slots.

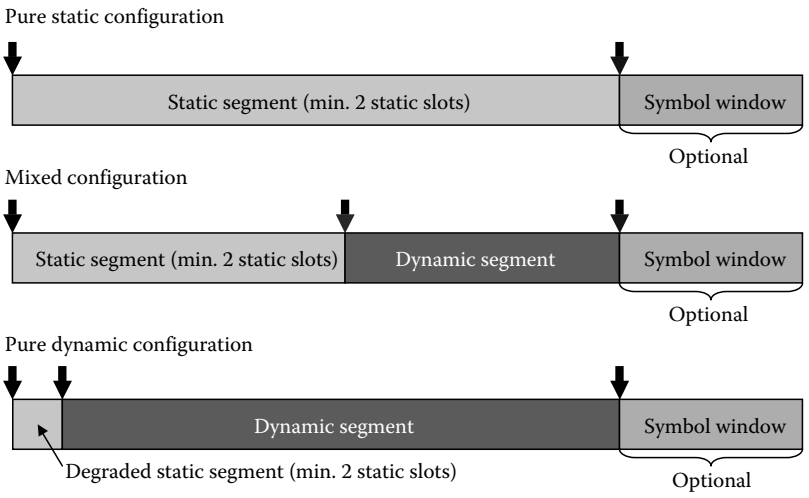


FIGURE 16.5 FlexRay configurations.

Considering the most likely application domains, mixed configurations will be dominant. Depending on the actual configuration, FlexRay can achieve a best-case bandwidth utilization of about 70% with the average utilization ranging around 60%.

16.3.2 Clock Synchronization

The media access scheme of FlexRay relies on a common and consistent view about time that is shared between all CCs in the network. It is the task of the clock synchronization service to generate such a time view locally inside each CC. For the detailed description of the clock synchronization, first the representation of time inside a CC is described. The physical basis for each time representation is the tick of the local controller oscillator. This clock signal is divided by an integer to form a clock signal called microtick. This is often done by a frequency divider implemented in hardware. An integer number of microticks form a time unit called macrotick. Minislots and static slots are set up as integer multiple of macroticks. The number of microticks that constitute a macrotick is statically configured. However, for adjustment of the local time, the clock synchronization service can temporarily adjust this ratio in order to accelerate or decelerate the macrotick clock. If, for instance, a clock is too fast, microticks are added to a macrotick in order to slow down the counting.

The clock synchronization service is a distributed control system that produces local macroticks with a defined precision in relation to the local macroticks of the other controllers of a network. The control system takes some globally visible reference events that represent the global time ticks, measures the deviation of the local time ticks to these global ticks, and computes the local adjustments in order to minimize the deviation of the local clock from the global ticks. Due to the distributed nature of the FlexRay system, no explicit global reference event exists. The only event, which is globally observable, is a frame transmission on the communication channel. The start of a transmission is triggered by the local time base of the sending controller. Each controller can collect these reference events to form a virtual global time base by computing a fault-tolerant mean value of the deviation between the local time and the perceived events.

The reasoning behind this approach is based on the assumption that a majority of local clocks in the network is correct. Correctness of a local clock is given, when the local clock does not deviate from every other by more than the precision value. A controller with locally correct clock sees only deviation values within the precision value. By computing the median value of deviations, the actual

temporal deviation of the local clock to the virtual reference tick is formed. Next, the local clock is adjusted such that the local deviation is minimized. Since all nodes in the network perform this operation, all local clocks move their ticks toward the tick of the virtual global time. The operation of observation, computing, and correction is performed in every communication cycle.

In case of wrong transmission times of faulty controllers on the communication channel, things get more complicated. Here, a special part of the fault-tolerant median value algorithm takes over. This algorithm uses only the best of the measured deviation values. All other values are discarded. This algorithm ensures that the maximum influence of a faulty controller to the virtual global time is strictly bound. Additionally, the protocol requires the marking of particular synchronization frames that can be used for deviation measurement. The reasoning behind this mechanism is twofold. The first reason is that it is used to pick exactly one frame from a controller in order to avoid monopolization of the global time by one controller with many transmit frames. The second reason is that particular controllers can be excluded from clock synchronization, either because the crystal is not trustworthy or more likely because there are system configurations in which a controller is not available.

In case of a faulty local clock, the faulty controller perceives only deviation values that exceed a particular value. This value is given by the precision value. This condition is checked by the synchronization service and an error is reported to the application. A specific extension of the clock synchronization services handles the compensation of permanent deviations of one node. In case such a permanent deviation is detected, a permanent correction is applied. The detection and calculation of such permanent deviations is executed less frequently than the correction of temporal deviations.

The error handling of the protocol follows a strategy, which identifies every problem as fast as possible, but keeps the controller alive as long as possible. Problem indicators are frames that are received outside their expected arrival intervals or when the clock synchronization does not receive a sufficient number of synchronization frames. The automatic reaction of the controller is to degrade the operation from a sending mode to a passive mode where reception is still possible. At the same time the problem is indicated to the application. The application can react in an application-specific manner to the detected problem. This strategy gives the designer of a system a maximum of flexibility for the design of the safety required by the application.

16.3.3 Start-Up

The preceding protocol description handled only the case of an already running system. To reach this state the start-up service is part of the protocol. Its purpose is to establish a common view on the global time and the position in the communication cycle.

Generally the start-up service has to handle two different cases. The coldstart case is a start-up of all nodes in the network, while the reintegration case means to integrate a starting controller into an already running set of controllers (see Figure 16.6).

During coldstart, the algorithm has to ensure that really a coldstart situation is given. Otherwise the starting controller might disturb an already running set of controllers. For this reason, the starting controller has to listen for the so-called listen timeout for traffic on the communication channel. In case no traffic is detected, the controller assumes a coldstart situation and starts to transmit frames for a limited number of rounds. In case another controller responds with frames that fit to the slot counter of the coldstart node, start-up was successful.

In case traffic is detected during the observation period, the controller changes into the reintegration mode. In this mode, the controller has to synchronize the slot counter with the frames seen on the channel. Therefore, the controller receives frames from the channel and sets the slot counter accordingly. For a certain period of time, the controller checks the plausibility of the received frames

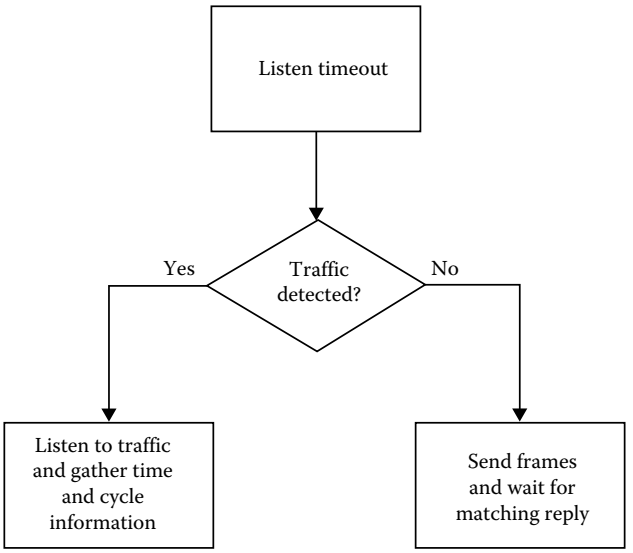


FIGURE 16.6 Start-up process.

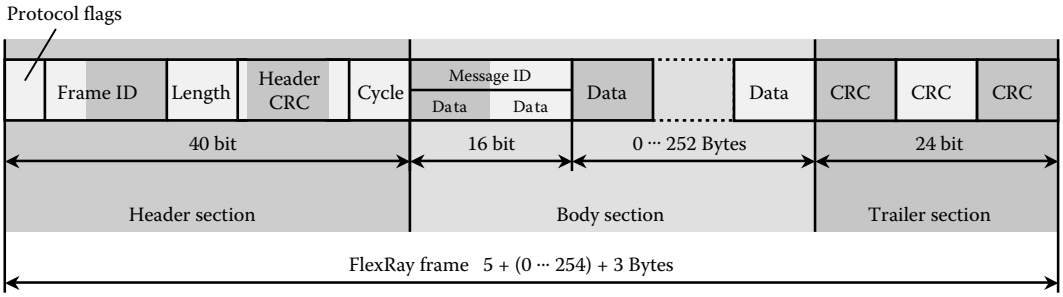


FIGURE 16.7 FlexRay frame format.

in relation to the internal slot counter. If there is a match, the controller enters the normal mode, in which active transmission of frames is allowed.

16.3.4 Coding and Physical Layer

The frame format for data transmission contains three sections: the header section, the payload section, and a trailer section (see Figure 16.7). The header contains protocol control information like the synchronization frame flag, the frame ID, a null frame indicator and the frame length, and a cycle counter. The payload section contains up to 254 bytes of data. In case the payload does not contain any data, the null frame indicator is set. A null frame is thus a valid frame that does not contain any payload data. It can, however, serve as an alive signal or can be used for clock synchronization purposes.

Optionally, the data section can contain a message ID, which identifies the type of information transported in the frame. The trailer section contains a 24 bit CRC that protects the complete frame.

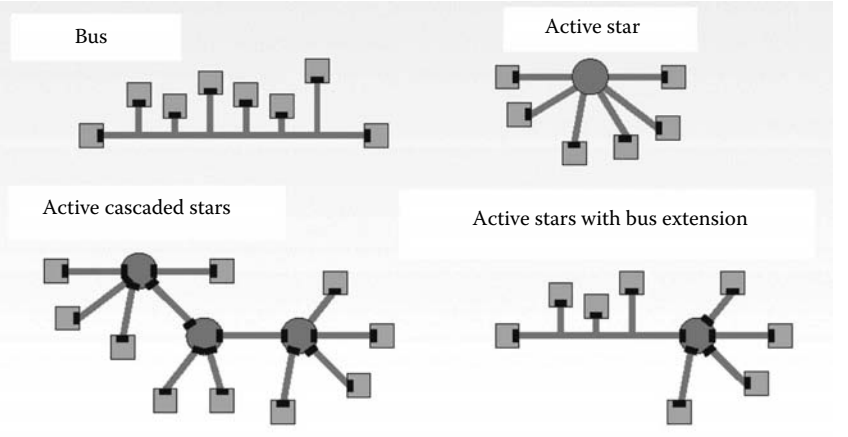


FIGURE 16.8 Topologies.

The existing FlexRay CCs support communication bit rates of up to 10 Mbps on two channels over an electrical physical layer. The physical layer is connected to the controller via a transceiver component.

This physical layer supports bus topologies, star topologies, cascaded star topologies, and bus stubs connected to star couplers as shown in Figure 16.8. This multitude of topologies allows a maximum of scalability and flexibility of electronic architectures in automotive applications.

Beside transformation of bit streams between the CC and the physical layer, the transceiver component also provides a set of very specific services for an automotive network. The major services are alarm handling and wakeup control. Alarm signals are a very powerful mechanism for diverse information exchange between a sender controller and receiver controllers. A sender transmits an alarm symbol on the bus parallel to alarm information in a frame. A receiver ECU receives the alarm information in the frame like normal data. Additionally, the CC receives the alarm symbol on the physical layer and indicates this symbol to the ECU. Thus the ECU has two highly independent indicators for an alarm to act on. This scheme can be used for the validation of critical signals like an air-bag fire command.

The second type of service provided by the symbol mechanism is the wakeup function. A wakeup service is required in automotive applications where electronic components have a sleep mode, in which power consumption is extremely reduced. The wakeup service restarts normal operation in all sleeping ECU components. In a network, the wakeup service uses a special signal that is transmitted over the network. In FlexRay, this function relies on the ability of a transceiver component to identify a wakeup symbol and to signal this event to the CC and the remaining components of the ECU to wake these components up.

16.3.5 Bus Guardian

The media access strategy completely relies on the cooperative behavior of every CC in a network. The protocol mechanisms inside a controller ensure this behavior to a considerable high level of confidence. However, for safety-relevant applications, the controller internal mechanisms do not provide a sufficiently high level of safety. An additional and independent component is required to ensure that no controller can disturb the media access mechanism of the network. This additional component is called bus guardian (BG).

In FlexRay, the BG is a component with an independent clock and is constructed such that an error in the controller cannot influence the guardian and vice versa. The BG is configured with its

own set of parameters. These are independent of the parameters of the controller, although both parameter sets represent the same communication cycle and slot pattern. During runtime, the BG receives synchronization signals from the controller in order to keep track with the communication cycle. Using its own clock the BG verifies those synchronization signals due to avoid being influenced by a faulty controller.

Typically, the BG will be combined with the transceiver component. Optionally, a central BG located inside a star coupler can be used.

16.3.6 Protocol Services

Application information is transmitted by the CC inside of frames. A frame contains one or more application signals. A controller provides an interface for frame transmission and reception that consists of buffers. A buffer consists of a control/status section and the data section. These sections have different semantics for receive and transmit frames and for static and dynamic slots.

The control section of transmit buffers for frames in the static segment contains the slot ID and the channel, in which the frame is transmitted. Once a buffer of this type is configured and the communication is started, the controller periodically transmits the data in the data section in the slot configured in the slot ID. When the application changes the data in the buffer, the subsequent transmission contains the new data. A special control flag allows modifying this behavior such that in case the application does not update the data in the buffer, a null frame is transmitted, signaling the failure to update other controllers. The control section of a receive buffer for frames in the static segment defines the slot ID and the channel from which the frame should be loaded into the buffer. The status section contains the frame receive status and the null frame indicator. One special flag indicates that a new frame has been received. It is important to note that the slot ID and the channel selection for slots in the static segment cannot be changed during operation.

Buffer status and control section for frames in the dynamic segment are similar to the buffers in the static segment. Differences result from the fact that the slot ID and the channel can be changed during normal operation and that multiple buffers can be grouped together to form a FIFO for frame reception from the dynamic section.

The CC provides a set of timers that run clocked by the synchronized time of the network. Several different conditions can be used to generate interrupts based on these timers. These interrupts are efficient means to synchronize the application with the timing on the bus.

16.3.7 FlexRay Current State

By the end of 2006, the first FlexRay series application was introduced in a car, proving the readiness for series application of the technology. Two further applications are scheduled for start of production in 2008 and 2009 in premium cars. In 2008, five silicon vendors provide microcontrollers with FlexRay interfaces in 16 and 32 bit architectures. Four silicon vendors provide physical layer chips.

The current version of the protocol specification is 2.1 rev A. As a result of the experience gained from the first series projects, the FlexRay Consortium continues to improve the FlexRay specification. These improvements focus on single channel version, standardization of the 2.5 and 5 Mbit transmission rate and a time-triggered master mode for the clock synchronization.

16.4 System Configuration

With the advent of the TDMA communication technologies and especially in the automotive application domain, the off-line configuration of networks gets increasingly important. Off-line

configuration means that the configuration parameters of the CCs are not generated during the runtime of the system, but are determined throughout the development time of the system. The processes for system development are not only mainly determined by the applied technology, but also driven by industry-specific technical or organizational constraints. In the following section the background for such a design process is described by first defining a model for the used information and second explaining the information processing.

The information model categorizes information into eight information domains. This categorization is comprehensive in the sense that each and every piece of development information belongs to one of the information domains (see Figure 16.9).

The functional domain defines entities called functions and the communication relations between them. Functions describe the functionality of the entire system, creating a hierarchy from very abstract high-level functions down to specific, tailored functions. A system normally comprises more than one function. In a vehicle, for example, the functional hierarchy would feature "chassis functions" on the top level, "steering functions" and "braking functions," the latter being even more detailed into "basic braking function," "anti-lock brake functions," and so on. A communication relation between functions or within a function starts at a sender function, connects the sender function with receiver functions, and has an assigned signal.

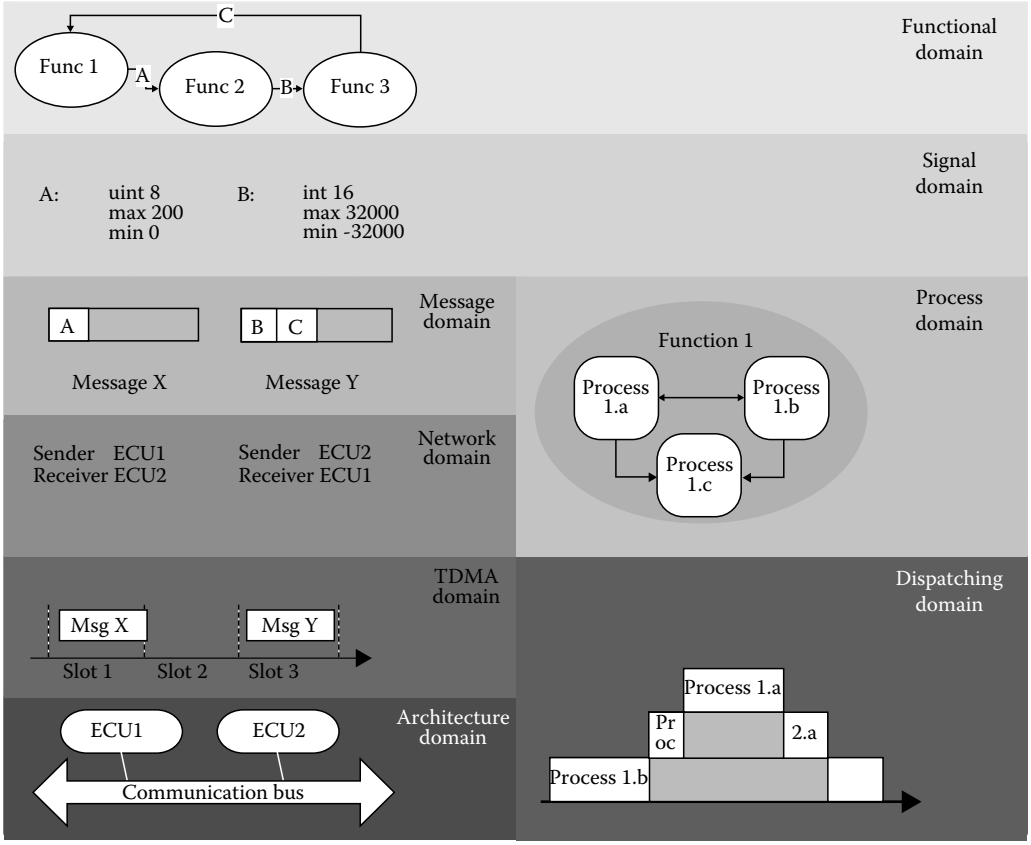


FIGURE 16.9 Information domains.

Signals are defined in the signal domain. A signal definition contains a signal name, the signal semantics, and the value range. Signals are assigned to messages. The message domain defines messages, i.e., packages of signals that should be transmitted together. A message has a name and a fixed setup of the signals it contains. The network domain determines which ECU will send a frame containing a message as well as which ECUs should receive this frame. The TDMA domain establishes the exact points in time when frames are transmitted. Finally, the architecture domain defines the physical structure of a system, with all ECUs, communication systems, and the connections of the ECUs to them.

The above six information domains are of importance for the entire system, i.e., for all ECUs of the systems. Hence, they are considered "global information."

Two additional domains complement the message, network, TDMA, and architecture domains. These two domains are the process domain and the dispatching domain. The process domain describes the software architecture of the system. It lists all processes and their interactions like mutual exclusion, as well as the assignment of processes to functions. Processes are information processing units of an application. They have timing parameters assigned to them that define the period and the time offset of their execution. The dispatching domain is the ECU counterpart of the TDMA domain. It determines the application timing, i.e., which process is executed at which point time. Implicitly, this also defines the preemption of processes, i.e., when a process is interrupted by the execution of another process. The latter two information domains feature information that is relevant for only one ECU. Hence, they are considered to be "local information" (local referring to one ECU rather than the entire system).

The categorization of information given by the information domain model leads the way to the development process. It is an organizational constraint in the automotive development processes that the knowledge related to the system to be developed is distributed among the process participants. These participants are typically the automobile manufacturer (OEM) and one or more suppliers. The OEM possesses the information on the intended system functions, the envisioned system architecture, and the allocation of functions to architectural components, i.e., to ECUs. Thus, the OEM's knowledge covers three of the six global information domains.

The supplier, on the other hand, is the expert on function implementation and ECU design. This means he provides the knowledge on the process domain, i.e., the software architecture underlying the function implementation. Each function of the system or each part thereof is implemented by a set of interacting processes. The functionality of the ECU does not only rely on the software architecture but also on the execution pattern. The supplier has to define the timing of each process executed in his developed ECU. Hence, the dispatching domain is the supplier knowledge as well.

With five of the eight information domains assigned, the open question is whose responsibility are the three remaining information domains. As described in the previous section, these domains—the message, the network, and the TDMA domain—define the communication behavior on the system level. They specify at which point in time, which message is transferred from which sending ECU to which receiving ECUs. These domains obviously affect all ECUs of the system; hence no single supplier should be able to define these domains. For this reason the DECOMSYS development process for collaborative system development between an OEM and several suppliers assigns the message, the network, and the TDMA domain to the OEM.

So far, each piece of development information has been assigned to a process participant, which results in a static structure. In the following appropriate dynamic structure, the development process will be described in brief.

The proposed OEM-supplier development process (Figure 16.10) takes a two-phase approach. In the first phase, the OEM has to cover all global aspects, subsequently the suppliers deal with the local aspects.

The process builds on the functional model of the system, which belongs to the functional domain and the signal domain, and the architectural model belonging to the architecture domain. The

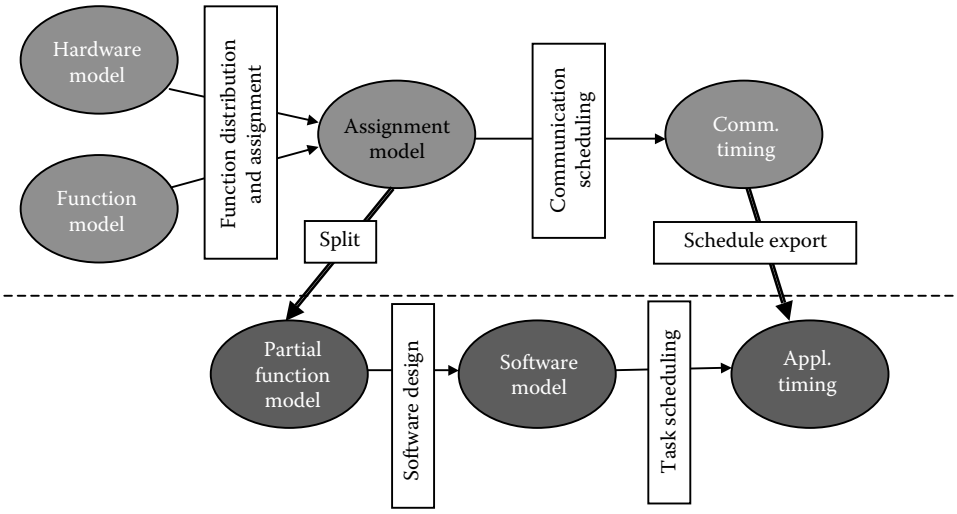


FIGURE 16.10 OEM-supplier development process.

functional model describes the functions of the system and their structure consisting of subfunctions. As subfunctions have to exchange information in order to provide the intended function output, the functional model inherently also defines the signals that are transferred from one subfunction to others. The functional model is complemented by the architectural model, which defines the system topology and the ECUs that are present in the system. The mapping between these two models results in a concrete system architecture for a particular system in a particular vehicle. This is described in the distributed functional model.

Based on the distributed functional model, the OEM performs communication scheduling. During this operation, signals are packed to messages, which in turn are scheduled for transmission at a specific point in time. Communication scheduling concludes the global design steps and thus the OEM's tasks.

The suppliers base their local design steps on the global information given by the OEM. The so-called split of the distributed functional model tells the supplier which functions or subfunctions the ECU has to perform, that is his responsibility. The supplier conducts software architecture design and creates the software model for each function or subfunction. The resulting list of processes is scheduled for the ECU taking into account local constraints as well as the global constraints defined in the communication schedule.

Note that for performing the local design, the supplier solely requires parts of the global information created by the OEM as well as his own knowledge on the ECU. In principle, the suppliers do not influence each other.

16.4.1 Development Models

The use of development models with a clear purpose and information content is the answer to the challenge of reuse of components for different car lines. Each model focuses on a certain type of information. The full picture of the system consists of these individual models and the mappings between them.

When developing a new car only those models affected by the differences between the previous version and the new vehicle have to be adapted while the other models remain unchanged. To be more specific, the reuse of system parts calls for the separation of the architectural and the functional

models. The functional model, i.e., the functions to be executed by the distributed system, is primarily independent of a specific car model and can be reused in different model ranges. The architectural model on the other hand, i.e., the concrete number of ECUs and their properties in a certain car, vary between model ranges.

It is decisive for a useful development process to allow the separate development of the functional model and the architectural model. At the same time the process must support the mapping of the functions to a concrete hardware architecture. The DECOMSYS OEM-supplier development process meets this requirement.

16.5 Standard Software Components

Standard software components within an ECU deliver a set of services to the actual application. The application itself can make use of these services without implementing any of them. For example, if a transport layer is part of the standard software components used in a project, the application does not need to take into account the segmentation and reassembly of data that exceed the maximum message size.

In order to reuse the application code in another project, the services offered by the standard software in the new project should be the same. If the standard software provides less functionality, the code has to be changed, as missing services have to be added.

In the optimal case the standardization effort covers all OEMs and suppliers. Only then reuse of existing code can be guaranteed, thus creating a win–win situation for all participants: The OEMs can purchase tested software that has proven its function and reliability in other projects—suppliers on the other hand have the possibility to sell this software, which they have created with considerable effort, to other OEMs.

16.5.1 Standardized Interfaces

Standard software components are not the only answer to the challenge of reuse. The standard software components and their standardized services must be complemented by standardized interfaces, through which the application software can access these services. Standardizing interfaces for software means to provide one operating system API for the software to access communication as well as other resources like ADCs. Similarly, standardized network interfaces allow the reuse of entire ECUs in different networks. The hardware of a distributed system can have a standardized interface represented by an abstract description of the network communication.

Standardization of software components as well as interfaces and thus a system architecture is not a competitive issue. Depending on where the interfaces are set, there is ample room for each participating company to use its strengths effectively to achieve its purpose. In our opinion, the real competitive issues are the functions in an ECU or the overall system functionality that is realized by the interaction of ECU functions. The special behavior of an electronic power-steering system as perceived by the car driver is mainly determined by the control algorithms and their application data, rather than by the type of communication interface used for integration.

With respect to standardization efforts, the industry is currently moving in the right direction. Initiatives like the open systems and the corresponding interfaces for automotive electronics (OSEK/VDX) Consortium [2], HIS [3], and many others attempt to standardize certain software components and interfaces. The field bus data exchange format (FIBEX) group that is now part of the Association for standardisation of automation and measuring systems (ASAM) Consortium [4] develops a standardized exchange format between tools based on XML, which is able to hold the complete specification of a distributed system. Many of these initiatives and projects are now united

in the automotive open system architecture (AUTOSAR) development partnership [5] with the goal to generate an industry wide standard for the basic software infrastructure.

References

1. www.flexray.com
2. www.osek-vdx.org
3. www.automotive-his.de
4. www.asam.de
5. www.autosar.de

17

LIN Standard

17.1 Introduction 17-1
17.2 The Need 17-2
17.3 History 17-2
17.4 Some LIN Basics 17-3
17.5 LIN Physical Layer 17-3
17.6 LIN Protocol 17-4
17.7 Design Process and Workflow 17-5
17.8 System Definition Process 17-6
17.9 Debugging 17-7
17.10 Future 17-7
Mentor Graphics LIN TOOL-CHAIN
17.11 LIN Network Architect 17-7
Requirement Capturing
17.12 LIN Target Package 17-9
17.13 LIN Spector—Test Tool 17-12
17.14 Summary 17-14
Acknowledgments 17-14
Bibliography 17-14
Additional Sources 17-14

Antal Rajnak
Mentor Graphics Corporation

17.1 Introduction

Local interconnect network (LIN) is much more than "just another protocol"! It is defining a straightforward design methodology, definition of tool-interfaces and signals-based application program interface (API) in a single package.

The LIN is an open communication standard, enabling fast and cost-efficient implementation of low-cost multiplex systems. It supports encapsulation for model-based design and validation, leading to front-loaded development processes which are faster, and more cost efficient than traditional development methods.

The LIN standard covers not only the definition of the bus-protocol, but expands its scope into the domain of application and tool-interfaces, reconfiguration mechanisms, and diagnostic services—thus offering a holistic communication solution for automotive, industrial, and consumer applications. In other words—Systems Engineering at its best—enabling distributed and parallel development processes.

Availability of dedicated tools to automate the design and System Integration process is a key factor for the success of LIN.

17.2 The Need

The car industry today is implementing an increasing number of functions in software. Complex electrical architectures using multiple networks, with different protocols are the norm in modern high-end cars. The software industry in general is handling software complexity through "best practices" such as

- Abstraction—hiding the unnecessary level of detail.
- Composability—partitioning a solution into a set of separately specified, developed, and validated modules, easily combined into a larger structure inheriting the validity of its components—without the need for revalidation.
- Parallel processes—state-of-the-art development processes such as the Rational Unified Process are based on parallel and iterative development were the most critical parts are developed and tested in the first iterations.

The automotive industry is under constant pressure to reduce cost and lead time, while still providing increasing amount of functionality. This must be managed without sacrificing quality. It is not uncommon today for a car project to spend half a billion U.S. dollars on development and perhaps as much as 150 million on prototypes. By shortening lead time the car maker creates benefits in several ways, typically both development cost and capital costs are reduced. At the same time an earlier market introduction creates better sales volumes and therefore better profit. One way of reducing lead time is by eliminating traditional prototype loops requiring full-size cars, rather relying on virtual development replacing traditional development and testing methods by computer-aided engineering.

To reduce development time while maintaining quality, a reduction in lead time must occur in a coordinated fashion for all major subsystems of a car such as body, electrical, chassis, and engine. With improved tools and practices for other subsystems and increasing complexity of the electrical system more focus must be placed on the electrical development process, as it may determine the total lead time and quality of the car. These two challenges—lead time reduction and handling of increased software complexity—will put growing pressure on the industry to handle development of electrical architectures in a more purposeful manner.

17.3 History

The LIN Consortium started in late 1998 initiated by five car manufacturers Audi, BMW, DaimlerChrysler, Volvo and Volkswagen, the tool manufacturer Volcano Communications Technology, and the semiconductor manufacturer Motorola. The workgroup focused on specification of an open standard for low-cost LINs in vehicles where the bandwidth and versatility of CAN is not required. The LIN standard includes the specification of the transmission protocol, the transmission medium, the interface between development tools, and the interfaces for application software programming. LIN promotes scaleable architectures and interoperability of network nodes from the viewpoint of hardware and software, and a predictable electromagnetic compliance (EMC) behavior. LIN complements the existing portfolio of automotive multiplex networks. It will be the enabling factor for the implementation of hierarchical vehicle networks, in order to gain further quality enhancement and cost reduction of vehicles. It addresses the needs of increasing complexity, implementation, and maintenance of software in distributed systems by provision for a highly automated tool chain.

The main properties of the LIN bus are

- Single master multiple slaves structure
- Low-cost silicon implementation using common universal asynchronous receiver transmitter (UART)/serial communication interface (SCI) interface hardware, an equivalent in software, or as pure state machine

- Self-synchronization without a quartz or ceramics resonator in the slave nodes
- Deterministic signal transfer entities, with signal propagation time computable in advance
- Signals-based API

A LIN network comprises one master and one or more slave nodes. The medium access is controlled by a master node—no arbitration or collision management in the slaves is required. Worst-case latency of signal transfer is guaranteed.

Today's LIN 2.1 standard is a matured specification, which has evolved through several versions (LIN 1.2, LIN 1.3, and LIN 2.0). LIN 2.0 took a major technology step by introducing new features such as configuration of slave nodes. The LIN 2.1 standard is reusing the experiences gained from use of LIN 2.0, and become the consensus of all major car OEMs. The LIN 2.1 standard completed definition of the diagnostic functionality (started in LIN 2.0) and made the configuration process more efficient.

LIN nodes produced today are based on LIN 1.2, LIN 1.3, LIN 2.0, and LIN 2.1. This means that backwards compatibility is an issue. The LIN standard solves this by ensuring strict backwards compatibility in LIN masters, and by introducing new features disjoint to older features. A LIN 2.1 master node can without much effort support heterogeneous clusters containing LIN 1.2 to LIN 2.1 slave nodes. New features added in later versions of the standard do not conflict (disjoint) with obsolete or older features.

17.4 Some LIN Basics

LIN is a low-cost, single wire network. The starting point of the physical layer design was the ISO 9141 standard. In order to meet EMC requirements the slew rates are controlled. The protocol is a simple master slave protocol based on the common UART format. In order to enable communication between nodes clocked by low-cost RC-Oscillators, synchronization information is transmitted by the master node on the bus. Slave nodes will synchronize with the master clock, which is regarded to be accurate. The speed of the LIN network is up to 20 kbps, and the transmission is protected by a checksum. The LIN Protocol is message-identifier based. The identifiers do not address nodes directly, but denote the meaning of the messages. This way any message can have multiple destinations (multicasting). The master sends out the message header consisting of a Synchronization Break (serving as a unique identifier for the beginning of the frame), a synchronization field carrying the clock information, and the message identifier, which denotes the meaning of the message.

Upon reception of the message identifier the nodes on the network will know exactly what to do with the message. One of the nodes sends out the message response and the others either listen or do not care. Messages from the master to the slave(s) are carried out in the same manner—in this case the slave task incorporated into the master node sends the response.

LIN messages are scheduled in a time-triggered fashion. This provides a model for the accurate calculation of latency times—thus supporting fully predictable behavior. Since the master sends out the headers, it is in complete control of the scheduling and is also able to swap between a set of predefined schedule tables, according to the specific requirements/modes of the applications running in the subsystem.

17.5 LIN Physical Layer

The transport medium is a single line, wired-AND bus being supplied via a termination resistor from the positive battery node (V_{BAT} Nominally 12 V). The bus line transceiver is an enhanced ISO 9141 implementation. The bus can take two complementary logical values: the dominant value with

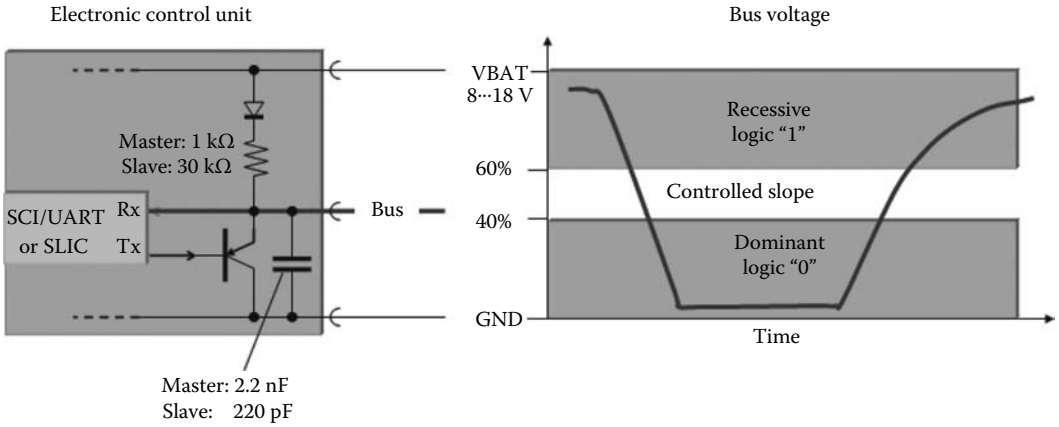


FIGURE 17.1 LIN logical level definitions. *Note*: The LIN specification refers to VBAT at the ECU connector.

an electrical voltage close to ground and representing a logical "0," and the recessive value, with an electrical voltage close to the battery supply and representing a logical "1" (Figure 17.1).

The bus is terminated by a pull-up resistor with a value of 1 kΩ in the master node and 30 kΩ in a slave node. A diode in series with the resistor is required to prevent the electronic control unit (ECU) from being powered by the bus in case of a local loss of battery. The termination capacitance is typically C_{Slave} = 220 pF in the slave nodes, while the capacitance of the master node is higher in order to make the total line capacitance less dependent from the actual number of slave nodes in a particular network. The maximum signalling rate is limited to 20 kbps. This value is a practical compromise between the conflicting requirements of high slew rates for the purpose of easy synchronization, and for slower slew rates for electromagnetic compatibility. The minimum band rate is 1 kbps—helping to avoid conflicts with the practical implementation of time-out periods.

17.6 LIN Protocol

The entities that are transferred on the LIN bus are frames. One message frame is formed by the header and the response (data) part. The communication in a LIN network is always initiated by the master task sending out a message header, which comprises the synchronization break, the synchronisation byte, and the message identifier. One slave task is activated upon reception and filtering of the identifier and starts the transmission of the message response. The response comprises one to eight data bytes and is protected by one checksum byte.

The time it takes to send a frame is the sum of the time to send each byte, plus the response space, and the interbyte spaces. The interbyte space is the period between the end of the stop bit of a byte and the start bit of the following byte.

The interframe space is the time from the end of a frame, until start of the next frame. A frame is constructed of a break followed by four to eleven byte-fields. The structure of a frame is shown in Figure 17.2.

In order to allow the detection of signaling errors the sender, of a message—as well as all receivers—are required to monitor the transmission. After transmission of a byte, the subsequent byte may only be transmitted if the received byte was correct. This allows a proper handling of bus collisions and time outs.

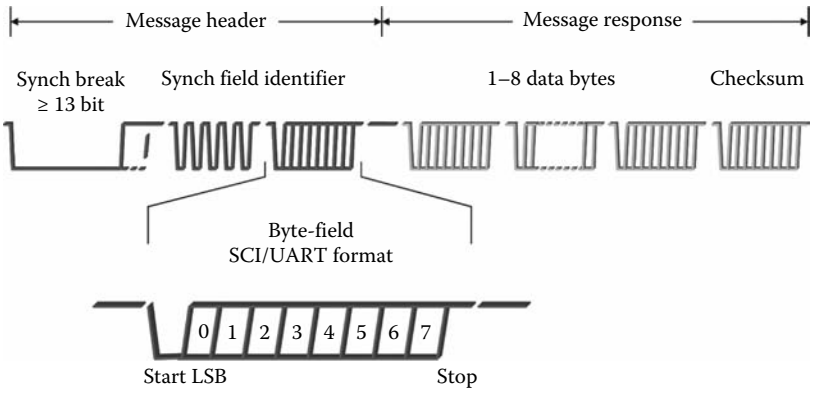


FIGURE 17.2 Frame structure.

Signals are transported in the data field of a frame. Several signals can be packed into one frame as long as they do not overlap each other. Each signal has exactly one producer, i.e., it is always written by the same node in the cluster. Zero, one, or multiple nodes may subscribe to the signal. A key property of the LIN protocol is the use of schedule tables. Schedule table makes it possible to assure that the bus will never be overloaded. They are also the key component to guarantee timely delivery of signals to the subscribing applications. Deterministic behavior is made possible by the fact that all transfers in a LIN cluster are initiated by the master task. It is the responsibility of the off-line synthesis/scheduling tool to assure that all frames relevant in a certain mode of operation are given enough time to be transferred.

17.7 Design Process and Workflow

Regardless of the protocol a network design process comprises of three major elements:

- Requirement capturing (signal definitions and timing requirements)
- Network configuration/design
- Network verification

The holistic concept of LIN supports the entire development, configuration, and validation of a network by providing definition of all necessary interfaces.

The LIN workflow allows for the implementation of a seamless chain of design and development tools enhancing speed of development and the reliability of the resulting LIN cluster.

The LIN configuration language allows description of a complete LIN network and also contains all information necessary to monitor the network. This information is sufficient to make a limited emulation of one or multiple nodes if it/they are not available. The LIN description file (LDF) can be one component used to generate software for an ECU which shall be part of the LIN network. The LDF is typically also used for verification, rest-bus simulation and as a configuration item (i.e., description of the LIN communication in a CM system). An API has been defined by the LIN Standard, to provide a uniform, abstract way to access the LIN network from applications. The syntax of an LDF is simple and compact enough to be handled manually, but use of computer-based tools is encouraged. Node capability files (NCF), as described in LIN Node Capability Language Specification, provides one way to (almost) automatically generate LDFs.

17.8 System Definition Process

Defining optimal signals packing and schedule table(s) fulfilling signalling needs in varying modes of operation, with consideration of capabilities of the participating nodes is called the System Definition Process. Typically, it will result in generation of the LDF file, written by hand for simple systems, or generated by high-level network design tools.

However, reusing existing, preconfigured slave nodes to create a cluster of them, starting from scratch is not that convenient. This is especially true if the defined system contains node address conflicts or frame identifier conflicts. The LIN Node Capability Language, which is a new feature in LIN 2.0, provides a standardized syntax for specification of off-the-shelves slave nodes. This will simplify procurement of standard nodes as well as provide possibilities for tools that automate cluster generation. The availability of such nodes is expected to grow rapidly. If accompanied by an NCF, it will be possible to generate both the LIN configuration file, and initialization code for the master node. Thus, true plug-and-play with nodes in a cluster will become a reality.

By receiving an NCF, with every existing slave node, the system definition step is automatic: Just add the NCF files to your project in the system definition tool and it produces the LDF file together with C code to configure a conflict free cluster. The configuration C code shall, of course, be run in the master node during start up of the cluster.

If you want to create new slave nodes as well, the process becomes somewhat more complicated. The steps to perform will depend on the system definition tool being used, which is not part of the LIN specification. A useful tool will allow for entering of additional information before generating the LDF file. (It is always possible to write a fictive NCF file for the non-existent slave node and thus, it will be included.)

An example of the intended workflow is depicted in Figure 17.3:

The slave nodes are connected to the master forming a LIN cluster. The corresponding NCF are parsed by the System Defining Tool to generate an LDF in the system definition process. The LDF is parsed by the System Generator to automatically generate LIN related functions in the desired nodes (the Master and Slave3 in the example shown in Figure 17.3). The LDF is also used by a LIN bus analyzer/emulator tool to allow for cluster debugging.

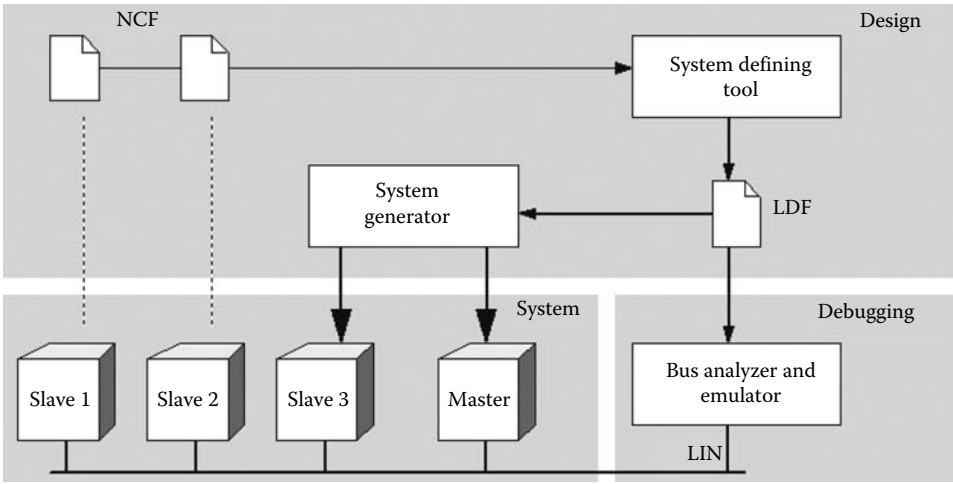


FIGURE 17.3 Workflow.

If the setup and configuration of any LIN cluster is fully automatic, a great step toward plug-and-play development with LIN will be taken. In other words, it will be just as easy to use distributed nodes in a LIN cluster as using a single node with the physical devices connected directly to the node.

It is worth noticing, that the generated LDF file reflects the configured network; any preexisting conflicts between nodes or frames must have been resolved before activating cluster traffic.

17.9 Debugging

Debugging and node emulation is based on the LDF file produced during system definition. Emulation of the master adds the requirement, that the cluster must be configured to be conflict-free. Hence, the emulator tool must be able to read reconfiguration data produced by the system definition tool.

17.10 Future

The driving ideas and the resulting technology behind the success of LIN—especially in the area of the structured approach towards the system design process—will most likely migrate to other areas of automotive electronics. Development of a Conformance Test Suite is under way, as well as specification for commercial vehicles using a 24 V electrical system. LIN itself will find its way to applications outside of the automotive world due to its low cost and versatility. The LIN Specification will evolve further to cover upcoming needs. For example, the future 42 V power supply will require a new physical layer. There will be a broad supply of components that are made for LIN. Because of high production volumes, these products can be used cost effectively in many applications, enhancing the functionality of vehicles in a more cost effective manner.

One example of a comprehensive tool-chain built around the open interface definitions of the LIN standard is presented below.

17.10.1 Mentor Graphics LIN TOOL-CHAIN

Process description: (Figure 17.4)

1. LIN network requirements are entered into the LIN network architect (LNA) synthesis tool.
2. Automatic frame compilation and schedule table generation by LNA.
3. LDF is generated by LNA.
4. LIN configuration generator (LCFG) Tool converts the LDF and the Private File (user information in an LCFG proprietary format) to target-dependent ".c" and ".h" source files.
5. Application code is compiled with target dependent configuration code and linked to the LIN Target Package (LTP) Library.
6. Analysis and emulation is performed with LIN Spector verification tool, using the generated LDF.

17.11 LIN Network Architect

LNA is a sophisticated synthesis tool built for design and management of LIN networks. Starting with the entry of basic data such as signals, encoding types and nodes, LNA takes the user through all stages of network definition.

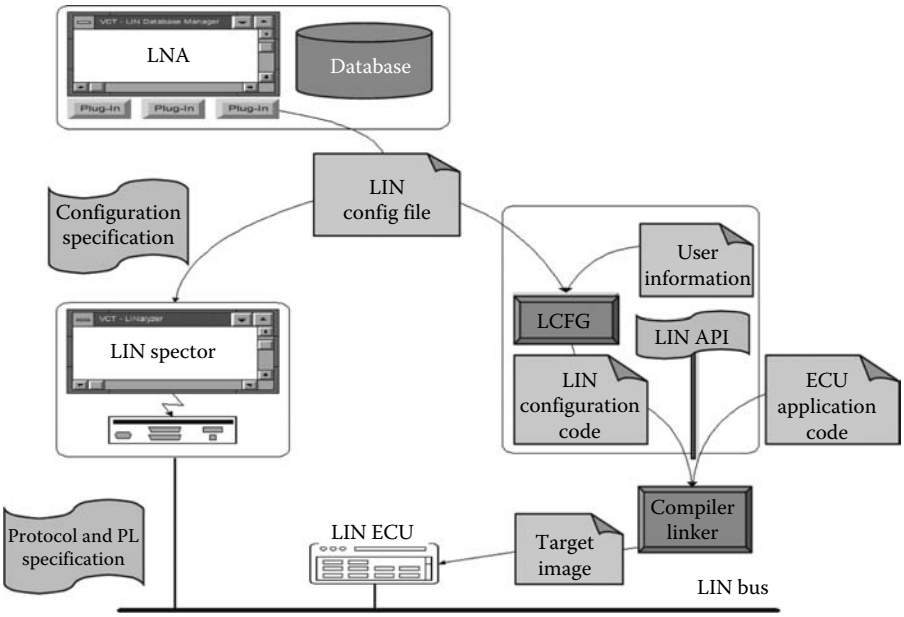


FIGURE 17.4 LIN configuration file (LDF) centric process flow.

17.11.1 Requirement Capturing

There are two types of data administered by LNA:

- Global objects (signals, encoding types, and nodes)
- Project-related data (network topology, frames, and schedule tables)

Global objects shall be created first, and can then be reused in any number of projects (Figure 17.5).

They can be defined manually, or imported by using a standardized XML input file (based on FIBEX rev.1.0). The tool can import data directly from the standardized NCF as well as from .dbc files. Comprehensive version and variant handling is supported.

The Systems Integrator combines subsets of these objects to define networks. Consistency check is being performed continuously during this process. This is followed by automatic packing of signals into frames (Figure 17.6).

The last task to complete is that of generating the schedule table in accordance with the timing requirements captured earlier in the process (Figure 17.7). The optimization is considering several factors such as bandwidth and memory usage.

Based on the allocation of signals to networks via node interfaces the tool will automatically identify gateway requirements between subnetworks, regardless whether LIN to LIN or LIN to CAN. The transfer of signals from one subnetwork to another will become completely transparent to the application of the automatically selected gateway node.

The tool uses a publish/subscribe model. A signal can only be published by one node but it can be received by any number of other nodes.

The max_age is the most important timing parameter defined by the timing model. This parameter describes the maximum allowed time between the generation and consumption of a signal involved in a distributed function (Figure 17.8).

Changes can be introduced in a straightforward manner, with frame definitions and schedule tables automatically recalculated to reflect the changed requirements.

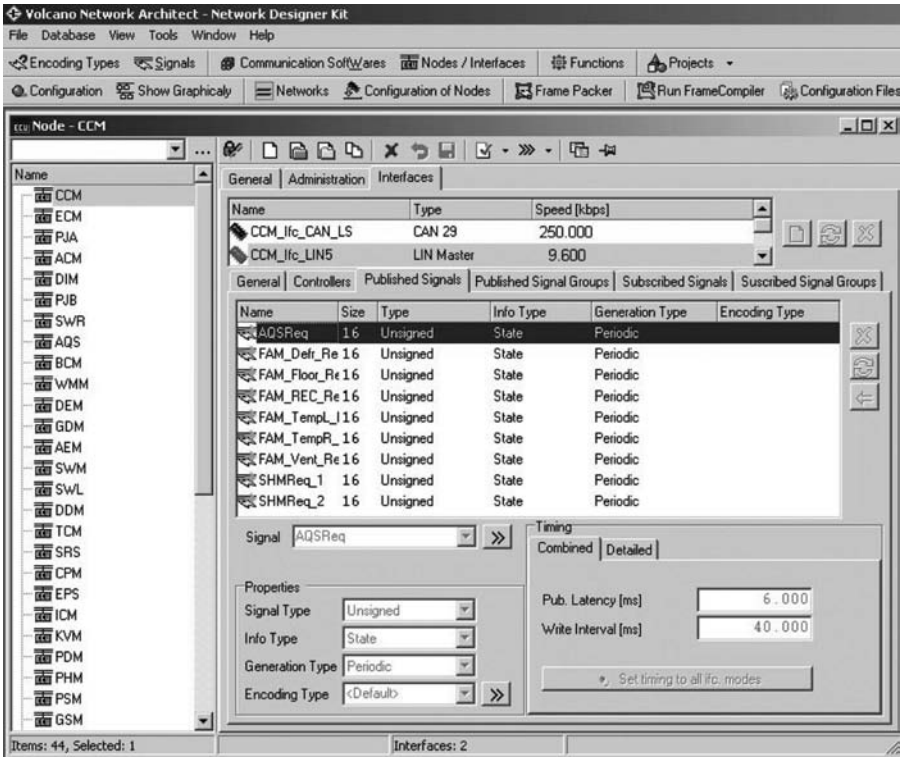


FIGURE 17.5 Definition of global objects in LNA.

When the timing analysis has been performed and the feasibility of the individual subnetworks has been established LDF files will automatically be created for each network (Figure 17.9).

Textual reports can be generated as well, to enhance the readability of information for all parties involved in the design, verification, and maintenance process.

17.12 LIN Target Package

The LTP represents the embedded software portion of the Volcano tool chain for LIN. The LTP is distributed as a precompiled and fully validated object library also including associated documentation and a command line configuration utility (LCFG) with automatic code generation capability, generating the configuration specific code and set of data structures.

Implementing the LTP with an application program is a simple process. The LDF created by the off-line tool contains the communication-related network information. In addition a Target File as an ASCII-based script defines low-level microcontroller information such as memory model, clock, SCI, and other node specifics to the LTP. These two files are run as input through the command line utility LCFG. It converts them both into target-dependant code usable by the microcontroller. The output contains all relevant configuration information formatted into compiler-ready C source code (Figure 17.10).

The target-dependent source code is added to the module build system along with the precompiled object library. After compilation the LTP gets linked to the application functionality to form the target image which is ready for download.

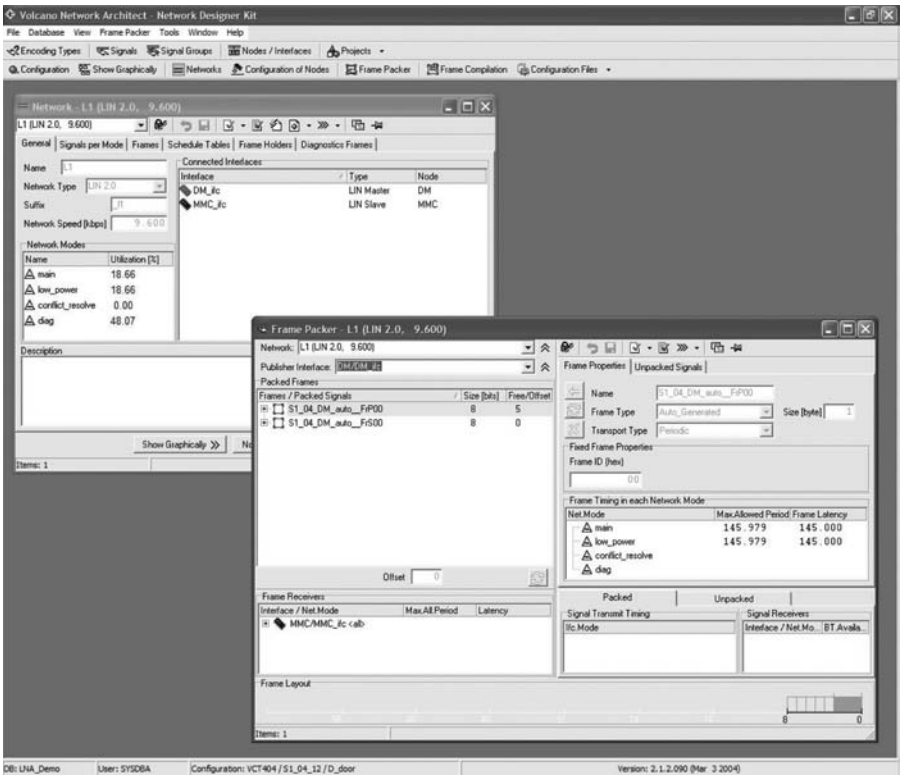


FIGURE 17.6 Network definition and frame packing.

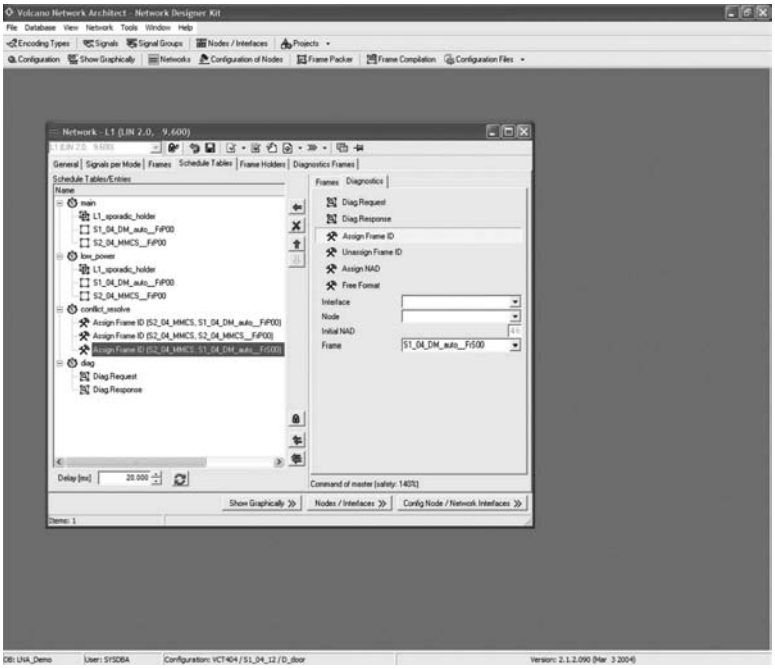


FIGURE 17.7 Manual and/or automatic schedule table generation.

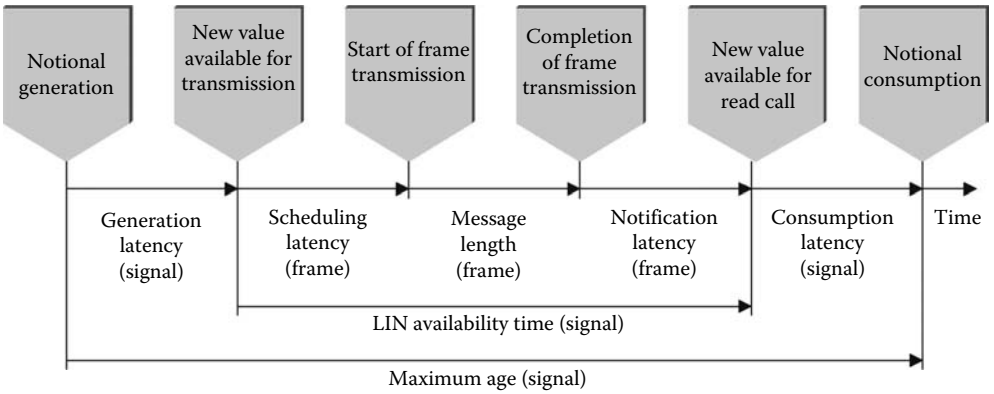


FIGURE 17.8 LIN timing model.

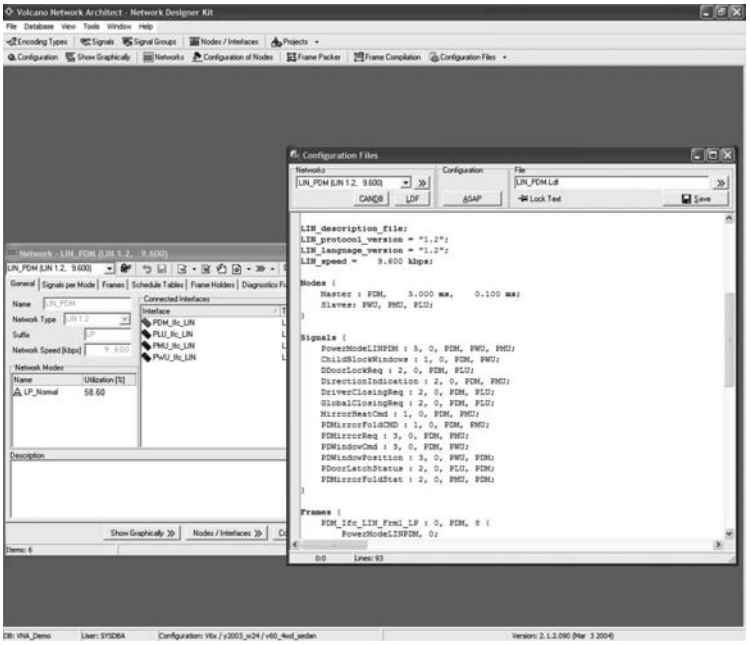


FIGURE 17.9 LDF file generation.

The application programmer can interface to the LTP and therefore to the LIN subnetwork through the standardized LIN API. API calls comprise of signal oriented read- and write calls, signal-related flag control but also node-related initialization, interrupt handling, and timed-task management.

The low-level details of communication are hidden to the application programmer using the LTP. Specifics about signal allocation within frames, frame IDs, and other are carried within the LDF so that applications can be reused simply by linking to different configurations described by different LDFs. As long as signal formats do not get changed a reconfiguration of the network only requires repeating the process described above resulting in a new target image without impact to the

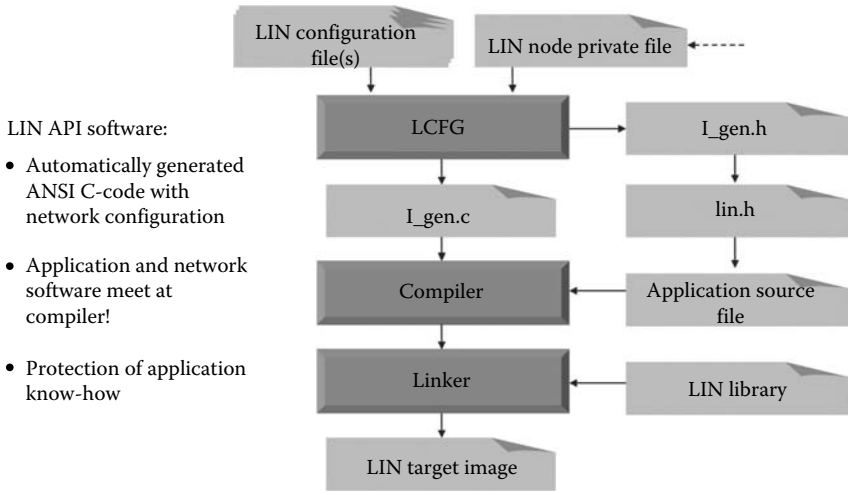


FIGURE 17.10 LIN target image building process.

application. When the node allows for reflashing the configuration can even be adapted without further supplier involvement allowing for end-of-line programming or after sales adaptations in case of service.

LTP's are created and built for a specific microcontroller and compiler target platform. A number of ports to popular targets are available and new ports can be made at the customer's request.

17.13 LIN Spector—Test Tool

LIN Spector is a highly flexible analysis and emulation tool used for testing and verification of LIN networks (Figure 17.11). The tool is divided into two components: an external hardware and a PC-based software package. Using a 32 bit microcontroller, the hardware portion performs exact low-level real-time bus monitoring (down to 10 μs resolution) while interfacing to the PC. Other connections are provided for external power, for bus and triggering connections. The output trigger is provided for connection to an oscilloscope, allowing the user to externally monitor the bus signalization.

Starting with LDF import, the tool allows for monitoring and display of all network signal data. Advanced analysis is possible with logical name and scaled physical value views. Full emulation of any number of nodes—regardless whether master or slave—is possible using LDF information. Communication logging and replay is possible, including the ability to start a log via logic-based triggers.

An optional emulation module enables the user to simulate complete applications or run test cycles when changing emulated signal values and switching schedule tables—all in real time. The functions are specified by the user via LIN Emulation Control files created in a C-like programming language, which can also be used to validate the complete LIN communication in a target module. Test cases are defined stressing bus communication by error injection on bit or protocol timing level.

Sophisticated graphical user interface panels can be created using the LIN GO feature within the test device (Figure 17.12).

These panels interface with the network data defined by the LDF for display and control.

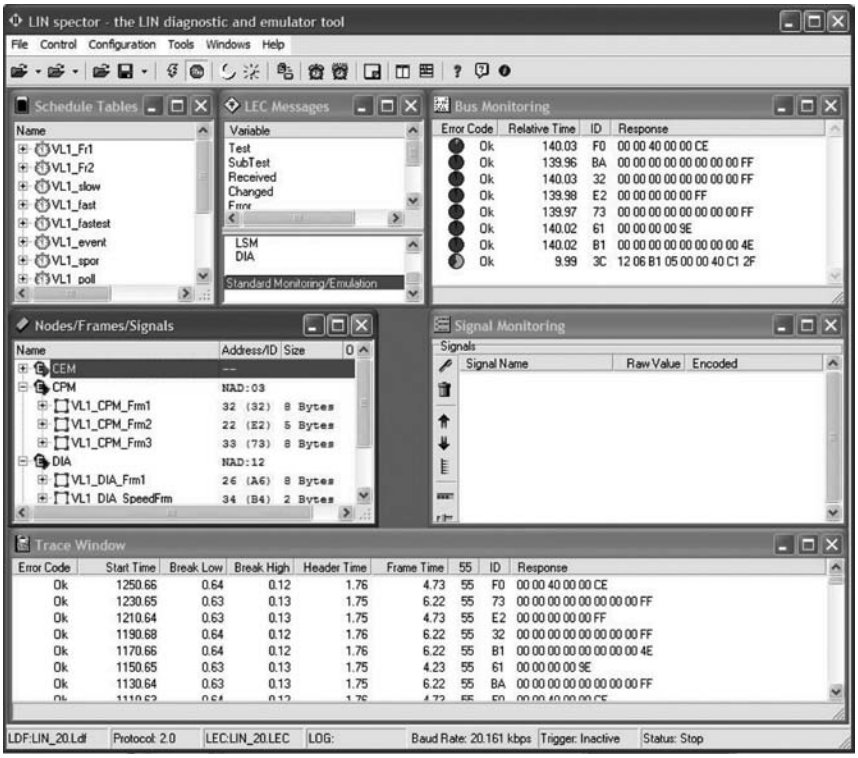


FIGURE 17.11 LIN Spector, diagnostics, and emulation tool for LIN.

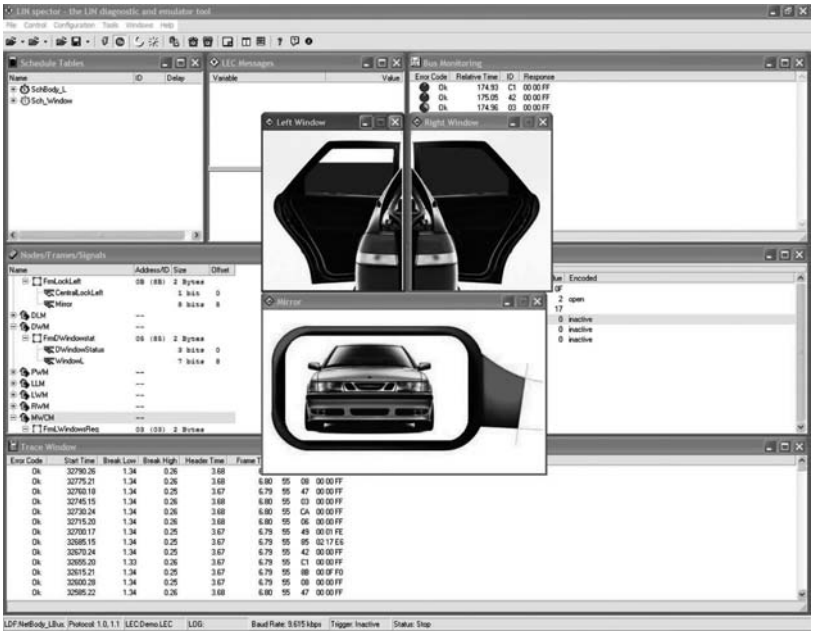


FIGURE 17.12 LIN GO—graphical objects.

17.14 Summary

LIN is an enabling factor for the implementation of a hierarchical vehicle network to achieve higher quality and reduction of cost for automotive makers. This is enabled by providing best practices of software development to the industry: abstraction and composability. LIN allows for reduction of the many existing low-end multiplex solutions and for cutting the cost of development, production, service, and logistics in vehicle electronics.

The growing number of car lines equipped with LIN and the ambitious plans for the next generation cars are probably the best prove for the success of LIN. The simplicity and completeness of the LIN specification combined with a holistic networking concept allowing for a high degree of automation has made LIN the perfect complement to CAN. The release of LIN 2.0 has further enhanced the reuse of components across car brands, and has added a higher degree of automated design capability by introduction of Node Capability Description Files and by defining mechanisms for reconfigurability of identical LIN devices in the same network.

Mentor Graphic Corporation (MGC) is offering the corresponding and highly automated tool chain to guarantee correctness by design. This shortens the design cycle, and allows for integration with higher level tools. LIN solutions provide a means for the automotive industry to drive new technology and functionality in all classes of vehicles.

Acknowledgments

I wish to thank Hans-Christian von der Wense of Freescale Semiconductor Munich; and Anders Kallerdahl, István Horváth, and Thomas Engler of Mentor Graphics Corp. for their contributions to this chapter.

Bibliography

LIN Consortium, LIN specification, version 2.1, www.lin-subbus.org, Nov 2006.

G. Reichart, LIN—a subbus standard in an open system architecture, *1st International LIN Conference 19*, September 2002, Ludwigsburg.

Road vehicles—Diagnostics systems—Requirement for interchange of digital information, International Standard ISO9141, 1st edn., 1989.

J. W. Specks, A. Rajnák, LIN—Protocol, development tools, and software interfaces for local interconnect networks in vehicles, *9th International Conference on Electronic Systems for Vehicles*, Baden-Baden, Oct 5/6, 2000, pp. 227–250.

W. Specks, A. Rajnák, The scaleable network architecture of the Volvo S80, *8th International Conference on Electronic Systems for Vehicles*, Baden-Baden, Oct 1998, pp. 597–641.

Additional Sources

The LIN-Specification Package and further Background information about LIN and the LIN-Consortium is available via the URL:
http://www.lin-subbus.org
Information about LIN products referred to in this chapter is available via the following URLs:
http://www.mentor.com/solutions/automotive
http://www.home.agilent.com/agilent/product.jspx?nid=-35137.685321.00& cc=US& lc=eng

18

Standardized System Software for Automotive Applications

18.1 Introduction 18-1
18.2 Hardware Architecture 18-2
18.3 OSEK/VDX 18-3
OSEK OS • OSEK NM • OSEK COM • OSEKtime OS • OSEKtime FTCom
18.4 HIS 18-7
Protected OSEK • I/O Library and Drivers • Flash Driver • CAN Driver
18.5 ISO 18-9
Transport Layer (ISO/DIS 15765-2.2) • Diagnostics—KWP2000 (ISO/DIS 14230-3)/UDS (ISO/DIS 14229-1)
18.6 ASAM 18-12
XCP—The Universal Measurement and Calibration Protocol Family
18.7 AUTOSAR 18-15
Application Software Architecture • System Software Architecture
18.8 JasPar 18-26
18.9 Summary 18-27
References 18-27

Thomas M. Galla
Elektrobit Austria GmbH

18.1 Introduction

In the last decade, the percentage of electronic components in today's cars has been ever increasing. According to Ref. [15], the S-Class Mercedes, for example, utilizes seven communication buses and 72 microcontrollers.

Since 1993, major automotive companies have been striving for the deployment of standard software modules in their applications as the potential benefits of using standard software modules are huge [14]. While the functional software heavily depends on the actual system and is a discriminating factor of competitive importance, this does not apply to the software infrastructure. Furthermore, with continuously shortened development cycles, especially in the electronics area, requirements arise concerning compatibility, reusability, and increased test coverage that can only be fulfilled by setting standards for the various system levels.

This trend has been a key motivation for the formation of several consortia like the "Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug/Vehicle Distributed Executive"

(OSEK/VDX)* consortium [5] in 1993–1994, the "Hersteller Initiative Software" (HIS)† group [3], the "Japan Automotive Software Platform Architecture" (JasPar) consortium [4] in 2004, the "Electronic Architecture and System Engineering for Integrated Safety Systems" (EASIS) project consortium [2] in 2003, and last but not least the "Automotive Open System Architecture" (AUTOSAR) consortium [11] in 2003.

This chapter provides an overview of today's state-of-the-art in standardization of automotive software infrastructures. The chapter is structured as follows: Section 18.2 provides short overview of the automotive hardware architecture. Section 18.3 provides information on the software modules specified by the German working groups OSEK/VDX. Section 18.4 describes the software modules defined by the HIS group. Section 18.5 illustrates the software modules standardized by the International Organization for Standardization (ISO). Section 18.7 deals with the AUTOSAR initiative. Section 18.8 focuses on the Japanese counterpart to AUTOSAR named JasPar, and Section 18.9 provides a short summary, concluding the chapter.

18.2 Hardware Architecture

The hardware architecture of automotive systems can be viewed at different levels of abstraction. On the highest level of abstraction, the "system level," an automotive system consists of a number of networks interconnected via gateways (see Figure 18.1). In general, these networks correspond to the different functional domains that can be found in today's cars (i.e., chassis domain, power train domain, body domain).

The networks themselves comprise a number of electronic control units (ECUs), which are interconnected via a communication media (see zoom-in on network A and D in Figure 18.1). The physical

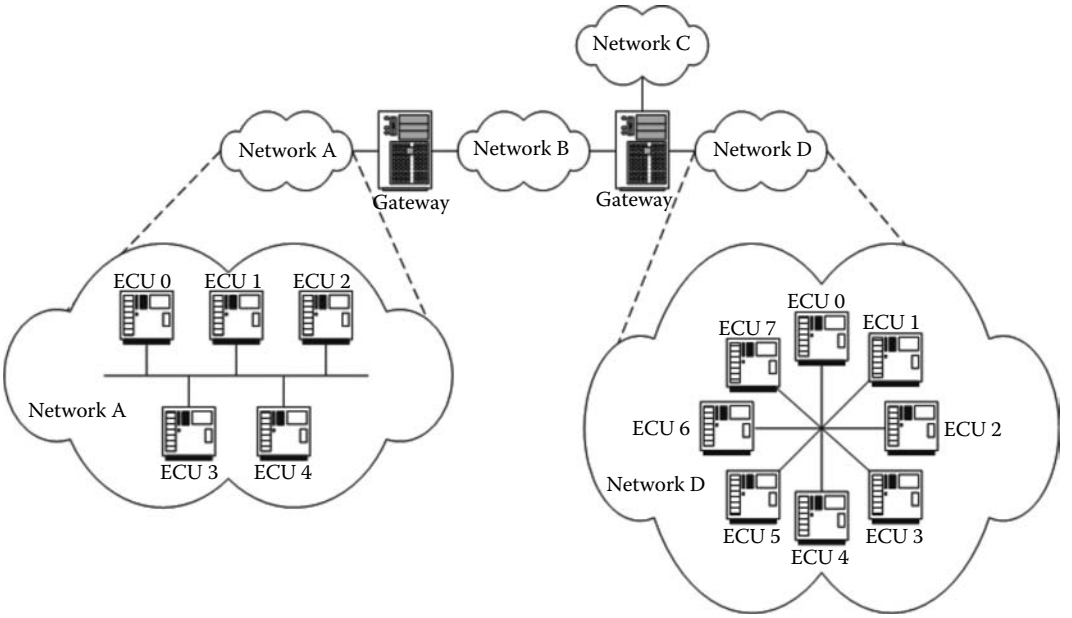


FIGURE 18.1 Hardware architecture—system and network level.

* Translated into English: "Open Systems and the Corresponding Interfaces for Automotive Electronics."
† Translated into English: "Manufacturers' Software Initiative."

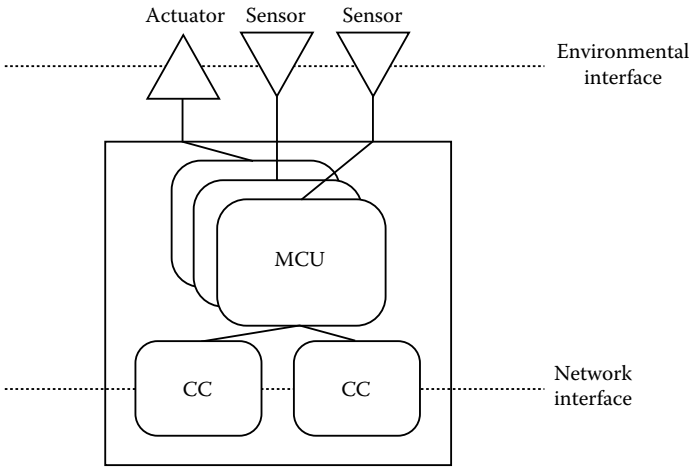


FIGURE 18.2 Hardware architecture—ECU level.

topology used for the interconnection is basically arbitrary; however, bus, star, and ring topologies are the most common topologies in today's cars. This "network level" represents the medium level of abstraction.

On the lowest level of abstraction, the "ECU level" (Figure 18.2), the major parts of an ECU are of interest. An ECU comprises one or more microcontroller units (MCUs) as well as one or more communication controllers (CCs). In most cases, exactly one MCU and one CC are used to build up an ECU. To be able to control physical processes in the car (e.g., control the injection pump of an engine), the ECU's MCU is connected to actuators via the analog or digital output ports of the MCU. To provide means to obtain environmental information, sensors are connected to the MCUs analog or digital input ports. We call this interface the ECU's environmental interface. The CC(s) facilitate(s) the physical connectivity of the ECU to the respective network(s). We call this interface of an ECU the ECU's network interface.

18.3 OSEK/VDX

The OSEK/VDX standard was the result of the endeavors of major car manufacturers and their suppliers to create a standardized software infrastructure for automotive electronics. This standard was initially designed for applications in the area of automotive body electronics or for the power train where autonomous control units build up a loosely coupled network. It comprises the following standardized components:

OSEK OS: The OSEK OS is an event-driven-operating system intended for hard real-time applications. OSEK provides services for task management, intertask communication by means of messages, events, resources, alarms, and interrupt handling.

OSEK NM: OSEK NM provides network management (NM) facilities, which take care of a controlled coordinated start-up and shutdown of the communication of multiple ECUs within a network.

OSEK COM: OSEK COM offers services to transfer data between different tasks and/or interrupt service routines (ISRs) residing on the same ECU (internal communication) or possibly being distributed over several ECUs (external communication).

OSEKtime OS: OSEKtime OS is a time-driven-operating system designed for minimal-operating system footprint and deployment in safety-related applications.

OSEKtime FTCom: OSEKtime FTCom is the fault-tolerant communication layer accompanying OSEKtime OS.

To minimize the memory footprint as well as the required central processing unit (CPU) time, each of these components is configured upon design time via a configuration file in OSEK implementation language syntax [10]. Using a generation tool, the appropriate data structures in the used programming language are created based on this configuration file. Thus, none of the components provides services for dynamic resource allocation (like task creation, memory allocation, and so on).

In the following, each of the listed OSEK/VDX components is described briefly.

18.3.1 OSEK OS

OSEK OS [30] is an "event-driven single chip real-time operating system," which is statically configured (as far as available tasks, used resources, etc.) and thus does not provide dynamic resource allocation. The OS itself defines three different processing levels: "interrupt level," "logical level" for the scheduler itself, and "task level". For the task scheduling, OSEK OS uses a "priority-based scheduling" strategy. Based on the OS configuration, this scheduling strategy is either "preemptive or non-preemptive." The main OS entities of execution are tasks in OSEK OS. Hereby, OSEK OS distinguishes between "basic tasks" (which do not use any blocking interprocess communication (IPC) constructs) and "extended tasks" (which are allowed to use blocking IPC constructs). The second entity of execution in OSEK OS are "ISRs." OSEK OS distinguishes between category 1 and category 2 ISRs. Whereas the former are not allowed to call operating system services, the latter are allowed to do so. Depending on the chosen scheduling strategy, an OSEK OS task can be in one of the states depicted in Figure 18.3.

Hereby, Figure 18.3a illustrates the task states for the extended task model (which exhibits an additional `Waiting` state), whereas Figure 18.3b depicts the tasks states for basic task model. Synchronization between extended tasks in OSEK OS is performed via "events." In case extended tasks can wait for an event, they remain in a `Waiting` task state until the event is set by some other extended task. Mutual exclusion between extended tasks is performed by means of "resources." To prevent deadlocks and priority inversion, OSEK OS implements the "priority ceiling protocol" [13] for resources. Communication between tasks in OSEK OS is performed by means of "messages." A detailed description of the OSEK communication primitives is provided in the section on OSEK COM (Section 18.3.3). For cyclic operations, OSEK OS provides the OS entities "counters" and

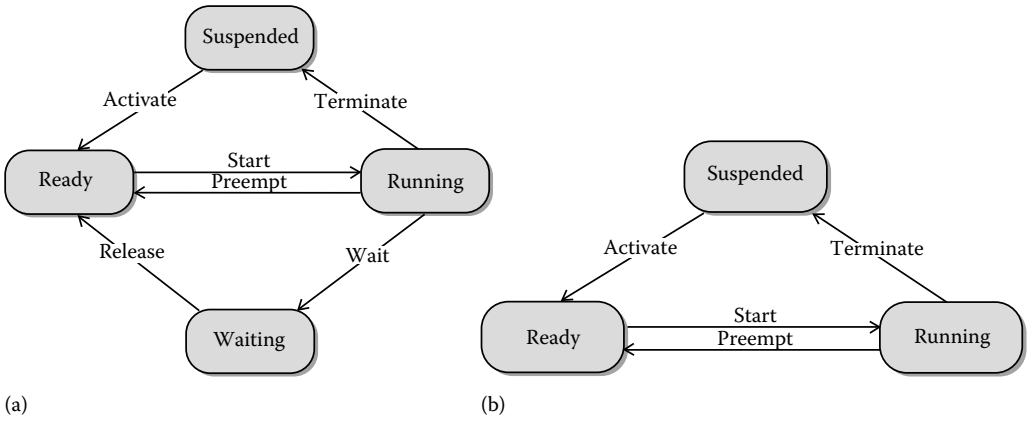


FIGURE 18.3 OSEK OS task states.

"alarms." Counters are represented by a counter value. Hereby, OSEK OS takes care of periodically incrementing this counter value. In case the counter value reaches a defined threshold, an alarm, which is linked to the counter, can be triggered. This alarm can, for example, set an event or activate a suspended task. To account for different modes of operations of an automotive real-time system, OSEK OS provides different "application modes." Each application mode owns its own subset of operating system entities (e.g., tasks, ISRs, etc.). During operation, a switch between these application modes is possible.

18.3.2 OSEK NM

NM in the automotive area handles the "controlled coordinated start-up and shutdown" of the communication of multiple ECUs within a network. The shutdown of the network (and the accompanying transitions of the ECUs into a low-power or even a power-down mode) is in general done to reduce the network's power consumption in situations, where the functions performed by the respective ECUs are not required (e.g., when the car is safely stored in a garage). Additionally, OSEK NM [21] provides (weakly-agreed) information on which ECUs within a network are participating in the OSEK NM. This service is called "node monitoring" in OSEK NM. To realize these two services, OSEK NM establishes a "logical ring" among all ECUs participating in OSEK NM. Along this ring, OSEK NM "ring messages" are passed between the participating ECUs. These ring messages contain information on whether the sending ECUs desires to perform a transition into a low-power sleep mode. In case all ECUs along the logical ring agree on this transition (i.e., no ECU objects), a coordinated transition into the sleep mode is performed. In case any ECU objects to this decision since it still requires network operation, a transition into the sleep mode is prevented. Wakeup of a sleeping network is triggered by any traffic on the network. Figure 18.4 depicts the state automaton of OSEK NM.

In case an ECU, which is currently not a member of the logical ring, wants to join the ring to participate in OSEK NM, the ECU's NM modules send a so-called "alive message" to introduce itself

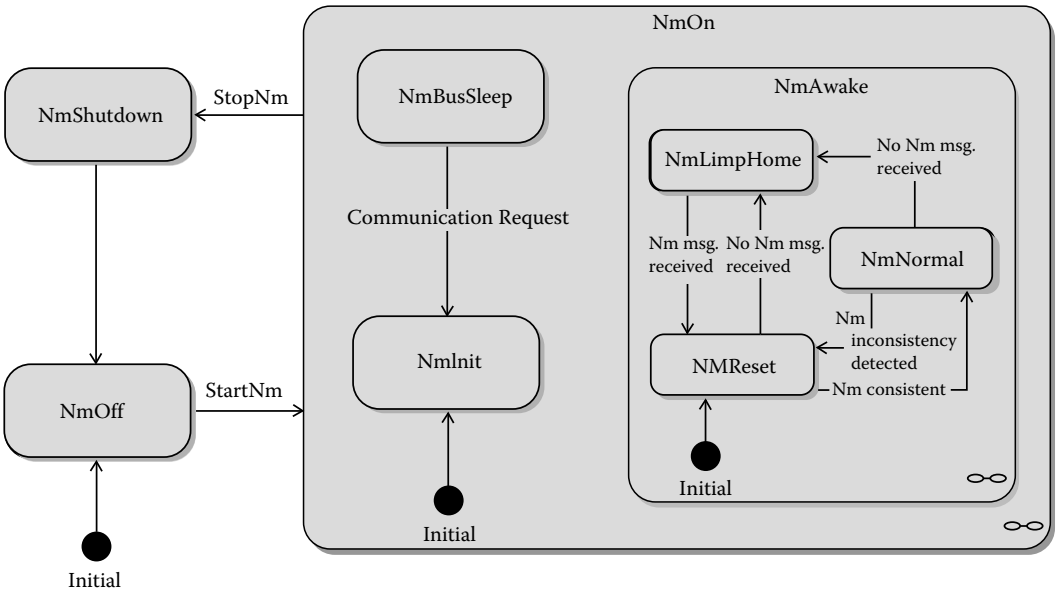


FIGURE 18.4 OSEK NM state automaton.

FIGURE 18.5 OSEK NM message format.

(substate `NmReset`). Once the new ECU has joined the ring, it participates in transmission of the ring messages (substate `NmNormal`). In case an ECU fails to receive any ring messages while residing in `NmNormal` state, the ECU NM module transits into `NmLimpHome` state, where the ECU sends "limp home messages" until ring messages are received again.

Figure 18.5 depicts the message format of OSEK NM's messages. A "source" and a "destination ID" field uniquely identify the sending and the receiving ECU in the network. The "OP code" field defines whether the message is a ring message, a limp home message, or an alive message. An optional data field may carry additional data from higher layers that this transferred along the ring.

18.3.3 OSEK COM

OSEK COM [9] provides a uniform communication environment for automotive control unit applications by offering services for data transfer among tasks and/or ISRs. Hereby, tasks and ISRs can be located on either the same ECU (internal communication) or on different ECUs (external communication). OSEK COM supports both "cyclic time-driven communication" and "event-driven communication." Additionally OSEK COM provides mechanisms for transmission and reception timeout monitoring (e.g., in case the transmission of a signal triggered by the call of an OSEK COM application programming interface (API) service does not take place within a statically configurable interval, a transmission timeout is signaled to the layers located on top of OSEK COM).

In case of external communication, the "interaction sublayer" of OSEK COM takes care of the representational issues of signals like byte ordering and alignment. On the sender's side, this layer converts signals from the byte order of the sending ECU into the network byte order. Furthermore, the interaction layer packs multiple signals into a single communication frame to reduce communication bandwidth consumption.

18.3.4 OSEKtime OS

OSEKtime OS [6] is designed as a statically configured time-driven single chip-operating system for distributed embedded systems. Just like OSEK OS, OSEKtime OS provides two entities of execution, namely, "ISRs" and "tasks." ISRs execute interrupt-related services. Tasks, however, are started at defined points in time and have one of three different states. In the `Running` state, the CPU is assigned to the task and its commands are executed. Only one task can reside in this state at any point in time while all other states can be adopted simultaneously by several tasks. The `Preempted` state is reached by a task that has been in the `Running` state and was preempted by another task that is to be activated. A task can leave the `Preempted` state only if the preempting task changes into the `Suspended` state. An inactive task that can be activated is in the `Suspended` state. These state changes are depicted in Figure 18.6.

The activation times of the tasks are stored in a "dispatcher table" before compile time. The dispatcher is the central component of the OSEKtime OS and is responsible for activating the tasks according to their activation times. The processing of the dispatcher table is done cyclically. A complete cycle of the dispatcher table is called a "dispatcher round." If a time-driven task is still in the state `Running` when the activation time of another time-driven task is reached, the first task passes over

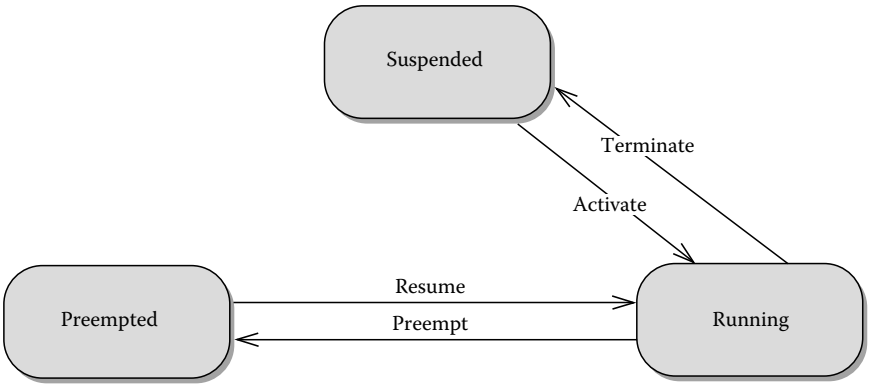


FIGURE 18.6 OSEKtime task states.

to the `Preempted` state and stays there until the interrupting task is finished. This kind of scheduling is called "stack-based scheduling"; no priorities are assigned to the tasks—only the activation times define the precedence between different tasks.

18.3.5 OSEKtime FTCom

OSEKtime FTCom [28] is the fault-tolerant real-time communication layer that accompanies OSEKtime OS. Similar to OSEK COM, OSEKtime FTCom can be used for both external and internal communication. In contrast to OSEK COM, OSEKtime FTCom solely supports "time-driven signal transmission". In case of external communication, the interaction sublayer of OSEKtime FTCom takes care of the representational issues of signals like byte ordering and alignment. On the sender's side, this layer converts signals from the byte order of the sending ECU into the network byte order. Furthermore, the interaction layer packs multiple signals into a single communication frame to reduce communication bandwidth consumption. Additionally in case of external communication, the "fault tolerance sublayer" of OSEKtime FTCom manages all fault tolerance issues, namely signal replication and signal reduction. On the sender's side, this layer replicates a single application signal and thus produces multiple signal instances. These signal instances are handed over to the interaction sublayer for byte order conversion and packing. Afterwards, the signal instances are transmitted via redundant communication paths. Hereby, temporal redundancy (multiple transmissions on a single communication channel) and spatial redundancy (transmission on multiple communication channels) can be distinguished. On the receiver's side, multiple signal instances are collected from the interaction sublayer and reduced to obtain a single application signal, which is handed on to the application layer.

18.4 HIS

The HIS working group has been formed by their members to "bundle their activities for standard software modules, process maturity levels, software test, software tools, and programming of control units. The common goal is to achieve and use joint standards." Different from the OSEK/VDX working group, HIS solely includes car manufacturers (OEMs) as members. The components standardized by HIS are listed in the following:

Protected OSEK OS: The protected OSEK OS specification defines additional mechanisms and services (in addition to the conventional OSEK OS) to protect separate applications running on the

same ECU from each other in the value (memory protection) and in the temporal (execution time monitoring) domain.

Input/Output (*I/O*) *library and drivers*: The HIS I/O library provides a standardized interface to access the respective low-level drivers.

Flash driver: The specification of the standardized HIS flash driver describes a standard interface of services required for flash programming.

Controller area network (*CAN*) *driver*: The HIS CAN driver provides data-link layer services for the transmission and the reception of frames on CAN [23,24] networks.

In the following, each of the listed HIS components is briefly described.

18.4.1 Protected OSEK

HIS defines extensions to the OSEK OS specification to protect separate application tasks running on the same ECU [18]. These extensions:

- Provide means to prevent different application tasks within the same ECU to corrupt each other's data.
- Monitor and restrict the execution time of each single application task.
- Exclusively assign operating system resources to a single application task and prevent all other application tasks from accessing these resources.

To fulfill these services, the protected OSEK OS, however, requires hardware support, namely, a memory protection unit and the existence of multiple processor modes (e.g., a privileged mode for the OS and a nonprivileged mode for the application tasks). For the monitoring and the restriction of the execution time of each task, an OS timer is assigned to each task to trace the task's execution time and abort the task in case the task exceeds its statically assigned execution time budget.

18.4.2 I/O Library and Drivers

The HIS I/O library [20] is intended to provide a standard interface to low-level drivers (e.g., for digital I/O). The I/O library supports both synchronous (i.e., the caller of the library function is blocked till the function completes) and asynchronous (i.e., the caller of the library function just triggers the function, and is asynchronously informed once the I/O operation has completed) operation. In case of asynchronous operation, the I/O library itself takes care of the required buffer handling to decouple the calling function from the called driver. The I/O library and the corresponding drivers provide the following API functions:

Init: This API function is used for the initialization of the controlled device. The function must be called prior to any other service.

Deinit: This API function is used to de-initialize the controlled device. The function has to be called upon releasing of the device (e.g., prior to termination of the application).

Read: This API function is used to read data from the controlled device. The semantics of this function is similar to the UNIX `read(2)` system call.

Write: This functions is used to write data to the controlled device. The semantics of this function is similar to the UNIX `write(2)` system call.

Ioctl: This function can be used to access other services provided by the respective driver, which cannot be mapped to the semantics of any of the previously described API functions. The semantics of this function is similar to the UNIX `ioctl(2)` system call.

As far as resource management is concerned, the HIS I/O library relies on the services provided by OSEK-operating systems. The I/O drivers [19] standardized by HIS are drivers for digital I/O, pulse width modulation (PWM), pulse width demodulation, analog/digital converters (ADC), watchdog timer, capture compare timers, EEPROM drivers, and flash drivers.

18.4.3 Flash Driver

The HIS standardized flash driver component [16] provides defined services (including a corresponding API) for the initialization of the flash driver component, the de-initialization of the flash driver component, the erasing of the flash memory, and the writing to the flash memory. Hereby, the job of the flash driver is to provide a logical abstraction from the physical organization of the flash memory (e.g., segmentation, banking, a.s.o.). The HIS flash driver is designed to be used as a driver for the HIS I/O library. Conventional application areas for the HIS flash driver are the flash programming services of XCP (see Section 18.6.1) and ISO diagnostics (see Section 18.5.2)

18.4.4 CAN Driver

The HIS CAN driver [17] is an implementation of a data link layer for CAN networks. The HIS CAN driver provides a uniform interface for the reception and the transmission of frames on the CAN network and thus abstracts from the peculiarities of the underlying buffer configurations of the respective CAN controller (full CAN vs. basic CAN a.s.o.). The main services provided by the CAN driver are the following:

Frame reception: The CAN driver retrieves received frames from the hardware buffers of the CAN controller and informs upper software layers about the frame reception.

Frame transmission: On request of higher software layers, the CAN driver transfers frames to be transmitted into the hardware buffers of the CAN controller and informs higher software layers about the successful transmission of these frames.

Wakeup service: If supported by the respective CAN controller, the CAN driver provides services to put the CAN controller in a special low-power mode (the sleep mode). In case a CAN controller transits from sleep mode (due to traffic on the communication media) back into the normal operation mode, the CAN driver informs the higher software layers about this event.

Error handling: In case fatal errors occur on the CAN communication media, the CAN driver informs the higher software layers about these errors.

Note, however, that the HIS CAN driver does not provide an I/O library compliant interface!

18.5 ISO

Some of the software layers used for CAN networks in today's cars have been standardized by the ISO. The most prominent of these software layers are the following:

Transport layer (*ISO/DIS 15765-2.2*): This layer provides "unacknowledged segmented data transfer (USDT)" to all higher layers and thus facilitates exchange of data amounts larger than the maximum transfer unit (MTU) of the underlying communication system.

Diagnostics—KWP2000 (*ISO/DIS 14230-3*) *and UDS* (*ISO/DIS 14229-1*): Both the Keyword Protocol 2000 (KWP2000) and the unified diagnostic services (UDS) define diagnostic services which "allow a tester device to control diagnostic functions" in an ECU via a serial data link.

18.5.1 Transport Layer (ISO/DIS 15765-2.2)

The ISO transport layer [25] provides USDT services to higher software layers (e.g., ISO diagnostics [see Section 18.5.2]) by facilitating the transmission of messages, whose length is greater than the MTU of the underlying communication system. On the sender's side, the transport layer will split such long messages into multiple segments, each small enough for the underlying communication system. On the receiver's side, the transport layer reassembles these segments again. As far as the frame format is concerned, the ISO transport layer distinguishes between single frames (SF), first frames (FF), consecutive frames (CF), and flow control frames (FC). Hereby, SFs are used for reduction of protocol overhead in case the amount of data to be transmitted does not exceed the MTU. In that case, only an SF is used to exchange this data. Otherwise an FF, which contains the total number of data bytes to be sent as well as the first few data bytes, is transmitted followed by one or more CFs, which then contain the remaining data bytes. For flow control reasons (i.e., to prevent the sender from outpacing the receiver), the receiver is allowed to send FCs at defined points in time. Figure 18.7 illustrates the frame exchange in an unsegmented (Figure 18.7a) and a segmented transmission (Figure 18.7b).

Hereby, the frame format depicted in Figure 18.8 is used by the ISO transport layer. The source and "destination address" fields are used to identify the sender and the receiver(s) of the TP frame. The "address type" field is used to define the semantics of the destination address. Here, ISO TP provides support for single ECU addresses (called "physical addresses" by the ISO TP specification) and multicast addresses (called "functional addresses" by the ISO TP specification). In case of multicast transmissions, only unsegmented data transfer is allowed in ISO TP.

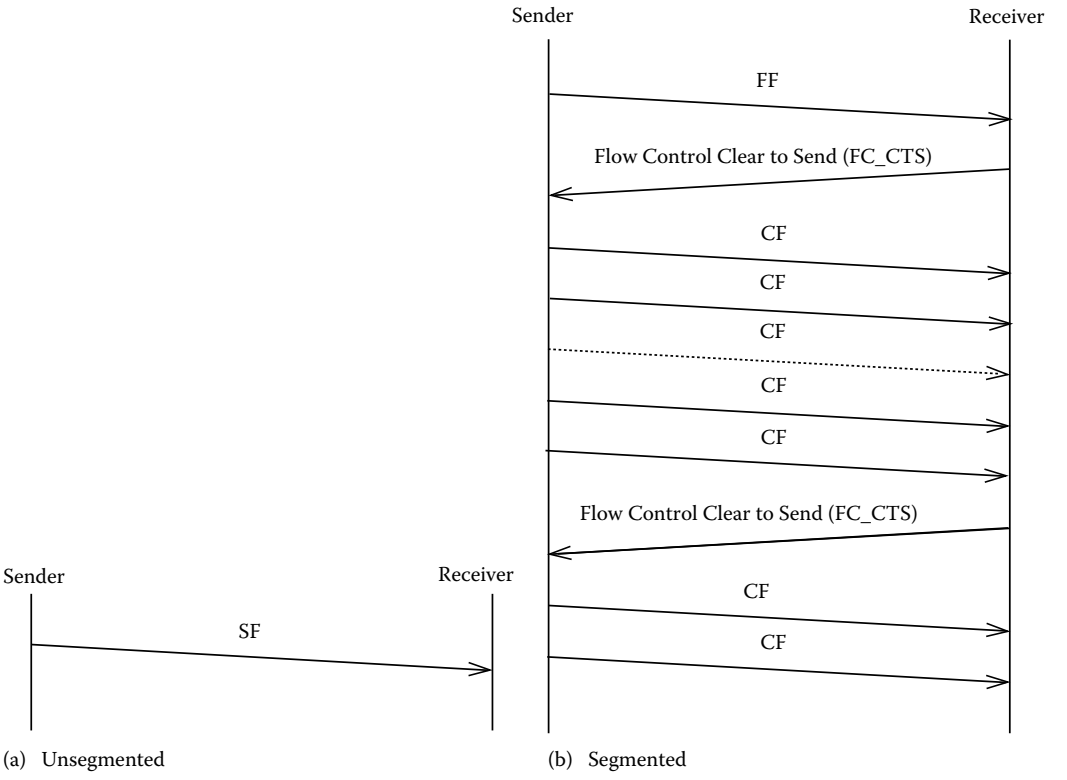


FIGURE 18.7 ISO transport layer—transmission sequence.

FIGURE 18.8 ISO transport layer—frame format.

The "protocol control information (PCI)" field is used to distinguish between the different frame types (i.e., SF, FF, CF, FC). Furthermore, the PCI field carries the "total number of data bytes" to be transmitted in case of a SF or an FF, a "sequence number" in case of a CF (to facilitate early detection of a loss of a CF), and more detailed information regarding the "type of flow control" in case of an FC.

18.5.2 Diagnostics—KWP2000 (ISO/DIS 14230-3)/UDS (ISO/DIS 14229-1)

The KWP2000 [22] and the UDS [26] define client/server diagnostic services, which allow a tester device (client) to control diagnostic functions in an ECU (server) via a serial data link. Hereby, the provided services, the encoding of the service identifiers (SID), and the encoding of the parameters are standardized by the KWP2000 and the UDS specification. For the exchange of information between the client and the server (and vice versa), both diagnostic protocols use the previously described transport layer. The services provided by KWP2000 and UDS can be grouped into the following six functional units:

Diagnostic management: This functional unit covers all services realizing diagnostic management functions between the client and the server. Examples for these kind of services are the services `startDiagnosticSession` for initiating a diagnostic session, `stopDiagnosticSession` for terminating a diagnostic session, `ecuReset` for resetting an ECU, and `readECUIdentification` for reading ECU identification information like the serial number or the programming date of the ECU.

Data transmission: This functional unit covers all services dealing with data exchange between the client and the server. Two prominent examples for these services are the services `readMemoryByAddress` and `writeMemoryByAddress`, which are used to read and write a memory range (defined by start address and the number of bytes to read/write) of the ECU.

Stored data transmission: This functional unit covers all services which are used to perform exchange of data, which are stored within the ECU (in a nonvolatile way). Examples for these kind of services are the service `readDiagnosticTroubleCodes` which is used to retrieve stored trouble codes from the ECU's error log as well as the service `clearDiagnosticInformation`, which is used to remove all entries from the ECU's error log.

I/O control: This functional unit covers all services which deal with input and output control functions. An example for a service contained in this group is the service `inputOutputControlByLocalIdentifier`, which can be used by the client to substitute a value for an input signal of the server.

Remote activation of routine: This functional unit covers all services which provide remote activation of routines within an ECU. Examples for services in this functional unit are the services `startRoutineByAddress` and `stopRoutineByAddress`, which are used to remotely invoke a routine in the server (ECU) and to terminate an executing routine in the server (ECU). Hereby, the routine to invoke or stop is identified by the routine's address.

Upload/download: This functional unit covers all services which realize upload/download functionality between the client and the server. Examples for services within this unit are the

`requestDownload` service, which gives the client the possibility to request the negotiation of a data transfer from client to server, the `requestUpload` service, which gives the client the possibility to request the negotiation of a data transfer from server to client, and the `transferData` service, which actually takes care of the data transmission between client and server.

In KWP2000 as well as in UDS, services are identified by the SID field, which is the first byte in a diagnostic message.* Based on this field, the layout of the remainder of the diagnostic message is completely different. The diagnostic server uses the SID to select the proper service requested by the client and interprets the remainder of the diagnostic message as parameters to the service request according to the SID.

Describing the format of the diagnostic messages for each value of SID is beyond the scope of this chapter. Detailed information on this topic can be obtained from the respective ISO specifications [22,26].

18.6 ASAM

The Association for Standardization of Automation and Measuring Systems (ASAM) started as an initiative of German car manufacturers with the goal to define standards for data models, interfaces, and syntax specifications for testing, evaluation, and simulation applications.

Apart from several data exchange formats like open diagnostic data exchange format (ODX), the functional specification exchange format (FSX), the meta data exchange format for software module sharing (MDX), and the fieldbus exchange format (FIBEX), ASAM defines the universal measurement and calibration protocol family (XCP), which is described in the next section.

18.6.1 XCP—The Universal Measurement and Calibration Protocol Family

The universal measurement and calibration protocol family (XCP) [29] is used for the following main purposes:

- Synchronous data transfer (acquisition and stimulation)
- Online calibration
- Flash programming for development purposes

Prior to describing these main operations of XCP though, we will focus on the protocol's internal structure. XCP itself consists of two main parts, namely, the "XCP protocol layer" and several "XCP transport layers," one dedicated transport layer for each underlying communication protocol (currently CAN, FlexRay, USB, TCP/IP, UDP/IP, and SxI are supported). Figure 18.9 illustrates this XCP protocol stack.

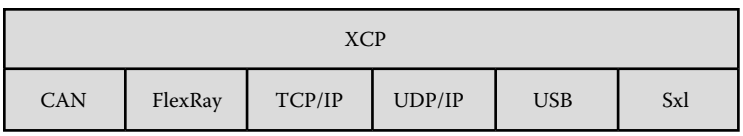


FIGURE 18.9 XCP protocol stack.

* This diagnostic message is payload from ISO TP's point of view.

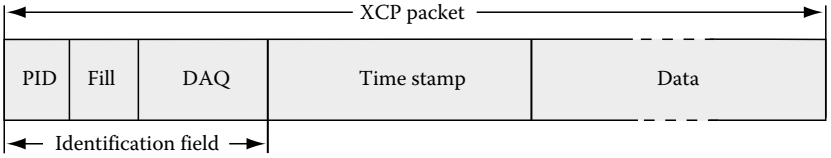


FIGURE 18.10 Structure of an XCP packet.

18.6.1.1 XCP Protocol Layer

The XCP protocol layer is the higher layer of the XCP protocol family. This layer implements the main operations of XCP, which are described in detail in Sections 18.6.1.3 through 18.6.1.5. The XCP protocol layer itself is independent of a concrete communication protocol (e.g., CAN, FlexRay, etc.). Data exchange on the XCP protocol layer level is performed by data objects called "XCP packets." Figure 18.10 illustrates the structure of an XCP packet.

An XCP packet starts with an "Identification Field" containing a "Packet identifier (PID)," which is used to establish a common understanding about the semantics of the packet's data between the XCP master and the XCP slave. The PID is thus used to uniquely identify each of the following two basic packet types and their respective subtypes:

Command transfer object (*CTO*) *packets*: This packet type is used for the transfer of generic control commands from the XCP master to the XCP slave and vice versa. It is used for carrying out "protocol commands (CMD)," transferring "command responses (RES)," "errors (ERR)," "events (EV)," and for issuing "service requests (SERV)."

Data transfer object (DTO) packets: This packet type is used for transferring synchronous data between the XCP master and the XCP slave device. "Synchronous data acquisition (DAQ) data" are transferred from the XCP slave to the XCP master whereas "synchronous data stimulation (STIM) data" are transported from the master to the slave.

The "DAQ Field" is used to uniquely identify the DAQ list (see Section 18.6.1.3) to be used for data acquisition or stimulation if the XCP packet is of DTO packet type. In case the packet is a CTO packet, the DAQ field is omitted.

The "Time-Stamp Field" is used in DTO packets to carry a time-stamp provided by the XCP slave for the respective data acquisition. The length of the time-stamp field may vary between 1–4 bytes depending on the configuration. In case the packet is of CTO type, the time-stamp field is omitted.

Command packets (CMD) are explicitly acknowledged on the XCP protocol layer by sending either a command response packet (RES) or an error packet (ERR). Event (EV) packets (i.e., packets informing the XCP master that a specific event has occurred in the XCP slave [e.g., an overload situation has occurred during data acquisition]), service request (SERV) packets (i.e., packets used by the slave to request certain services from the master [e.g., a packet containing text that is to be printed by the master]) and data acquisition packets (DAQ, STIM) are sent asynchronously and unacknowledged at the XCP protocol layer. Therefore, it may not be guaranteed that the master device will receive these packets when using a nonacknowledged transportation link like UDP/IP.

18.6.1.2 XCP Transport Layers

The protocol layer described in the previous sections is independent from the underlying communication protocol. To be able to use XCP on top of different communication protocols, the XCP specification defines multiple XCP transport layers that perform the packing of the protocol independent XCP packets into frames of the respective communication protocol, by adding an XCP header containing a "node address field" used to identify the destination ECU, a "counter field" used for

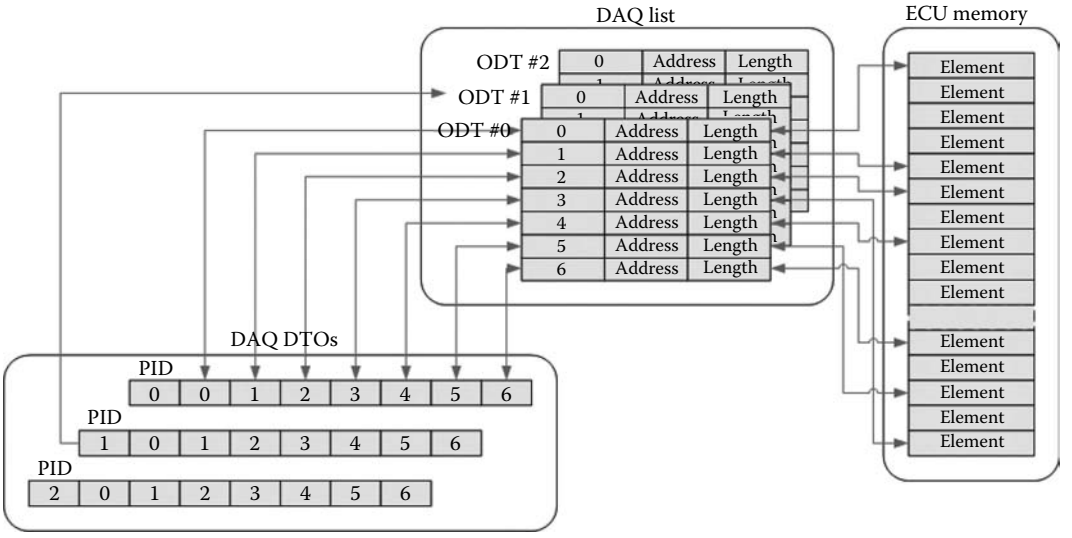


FIGURE 18.11 Structure of a DAQ list.

sequence numbering of the XCP packets and a "length field" defining the length of the XCP packet. Depending on the actual communication protocol used, some of these header fields might be missing (e.g., in case TCP/IP is used as communication protocol, the node address is omitted, as IP's addressing scheme is used).

18.6.1.3 Synchronous Data Transfer

The synchronous data transfer feature of XCP allows for a data exchange between XCP master and XCP slave that is performed synchronous to the XCP slave's execution. This exchange is carried out by using "DTOs," which are transferred via DTO packets. The memory regions of the XCP slave's memory that are the source or the destination of the transfer are linked to the DTO by so-called "object description tables (ODTs)." A sequence of one or more ODTs are grouped into a so-called "data acquisition (DAQ) list" (see Figure 18.11).

Hereby the DAQ-DTO contains a packet identification field (PID), which is used to link the DAQ-DTO to the respective ODT (PID field matches ODT number). For each element within a DAQ-DTO, a corresponding ODT entry is present in the ODT, which references a specific part in the ECU's memory by the attributes address and length. Upon processing of the ODT, the ODT entries are used to transfer element from the ECU's memory into the corresponding element in the DAQ-DTO (in case of data acquisition) and vice versa (in case of data stimulation).

DAQ lists can either be statically stored in an XCP slave ECU's memory or dynamically allocated by the XCP master via special protocol command packets.

18.6.1.4 Online Calibration

For the online calibration feature of XCP, the slave's physical memory is divided into so-called "sectors," which reflect the memory's size and limit constraints when reprogramming/erasing parts of the memory. This division into sectors thus describes the physical layout of the XCP slave's memory.

The logical layout is described by dividing the memory into "segments." This division does not need to adhere to the physical limitations of the division into sectors. Each segment can further consists of one or multiple pages, where at any given instance in time only one page of a segment is accessible to

the ECU. This page is called the active page for the ECU's application. The same holds true for XCP itself as well (active XCP page). XCP ensures that concurrent access to the same page by the ECU's application and XCP itself is prevented.

Via XCP CMD packets, the XCP master can instruct the XCP slave to switch active pages of the same segment (i.e., to make a page which has previously not been the active one the new active page) and to copy data between pages of the same or of different memory segments.

This way the calibration data used for the slave ECU's control loops, for example, can be altered upon run-time under control of the XCP master.

18.6.1.5 Flash Programming

To facilitate the exchange of the current image of the application program in the slave ECU's nonvolatile memory, XCP defines special commands (exchanged via XCP command packets) for performing flash (re-)programming. The following list contains a short description of the main commands used in flash programming:

PROGRAM_START: This indicates the beginning of a programming sequence of the slave ECU's nonvolatile memory.

PROGRAM_CLEAR: This clears a part of the slave ECU's nonvolatile memory (required before programming new data into this part of memory).

PROGRAM_FORMAT: It is used to specify the format for the data that will be transferred to the slave ECU (e.g., used compression format, used encryption algorithm, etc.).

PROGRAM: This command, which is issued in a loop as long as there is still data available that need to be programmed, is used to program a given number of data bytes to a given address within the slave ECU's nonvolatile memory.

PROGRAM_VERIFY: This verifies that the programming process has completed successfully by, for example, calculating a checksum over the programmed memory area.

PROGRAM_RESET: This indicates the end of the programming sequence and optionally resets the slave ECU.

By means of these flash programming XCP commands, a flash download process for the ECU's development stage can be implemented. In-system flash programming in series cars, however, is usually performed via the diagnostics protocol (see Section 18.5.2).

18.7 AUTOSAR

The development partnership AUTOSAR is an alliance of OEM manufacturers, major supplier companies, and tool vendors working together to develop and establish an open industry standard for an automotive electronic architecture, which is intended to serve as a basic infrastructure for the management of (soft and hard real-time) functions within automotive applications.

Thus far, AUTOSAR versions 1.0, 2.0, 2.1, and 3.0 have been released. For versions 1.0 and 2.0, a dedicated proof-of-concept implementation has been performed and integrated in a sample application.

The AUTOSAR software architecture makes a rather strict distinction between application software and basic or system software. While the "basic (or system) software" provides functionality like communication protocol stacks for automotive communication protocols (e.g., FlexRay [12,27]), a real-time-operating system, and diagnostic modules, the "application software," comprises all application-specific software items (i.e., control loops, interaction with sensor and actuators, etc.). This way, the basic or system software provides the foundation, the application software is built upon.

The so-called "*Runtime Environment* (`Rte`)" acts as an interface between application software components and the system software modules as well as the infrastructure services that enable communication to occur between application software components.

18.7.1 Application Software Architecture

Application software in AUTOSAR consists of "application software components," which are ECU and location independent and "sensor–actuator components" that are dependent on ECU hardware and therefore location dependent. Whereas instances of application software components can easily be deployed to and relocated among different ECUs, instances of sensor-actuator components must be deployed to a specific ECU for performance/efficiency reasons.

Application software components as well as sensor–actuator components are interconnected via so-called connectors. These connectors represent the exchange of signals or the remote method invocations among the connected components.

18.7.1.1 Runnable Entities

Software components are composed of one or more "runnable entities," which are executed in the context of operating system tasks. These runnable entities are invoked by a dedicated system software module named "*Runtime Environment* (`Rte`)." Depending on whether or not runnable entities have wait-points, AUTOSAR distinguishes between "category 1 runnable entities" (without wait-points) and "category 2 runnable entities" (with wait-points). The former can be assigned to basic tasks of the operating systems (which are not allowed to invoke any blocking system calls), whereas the latter have to be mapped to extended tasks (which are allowed to use blocking system calls).

18.7.1.2 Communication

To enable communication between different software components in AUTOSAR, the components are equipped with a communication interface, that consists of several ports. Hereby, ports have defined interface types. AUTOSAR distinguishes between "sender–receiver port interfaces," which provide message passing semantics, and "client–server port interfaces," which provide remote procedure call semantics. AUTOSAR further distinguishes between the following domains of communication: "intratask communication" (communication between runnable entities of components that are assigned to the same task on the same ECU), "intertask communication" (communication between runnable entities of components that are assigned to different tasks on the same ECU), and "inter-ECU communication" (communication between runnable entities of components that are assigned to different tasks on different ECUs). Each of the previously described communication semantics is supported for these three communication domains. As far as multiplicity is concerned, AUTOSAR supports 1:1, 1:n, and n:1 communication for sender–receiver port interfaces and 1:1 and *n*:1 communication (multiple clients, single server) for client–server port interfaces.

18.7.2 System Software Architecture

In addition to the application software components, AUTOSAR also defines a layered architecture of system software modules [7], which provides the basic platform for the execution of the application software components. Figure 18.12 provides a coarse grained overview of the major layers of these system software modules.

Microcontroller abstraction layer: This is the lowest software layer of the system software in AUTOSAR. This layer contains software modules, which directly access the microcontroller's internal

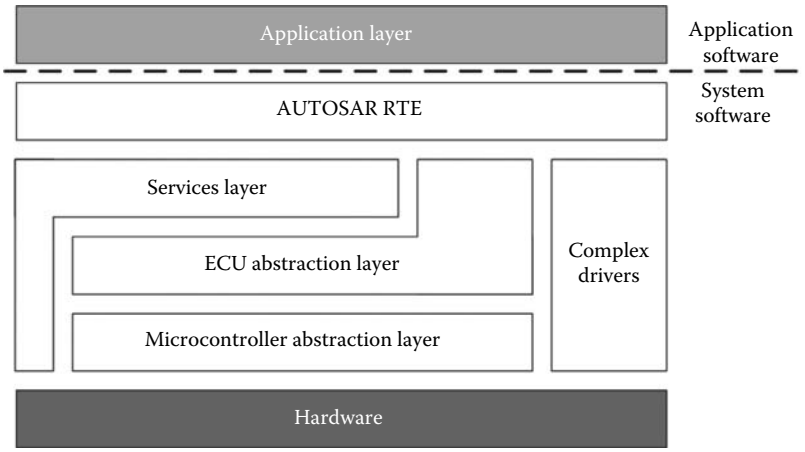


FIGURE 18.12 AUTOSAR—system software layered architecture.

peripheral devices as well as memory mapped external devices. The task of this layer is to make higher software layers independent of the microcontroller type.

ECU abstraction layer: This layer interfaces with the software modules of the microcontroller abstraction layer and additionally contains drivers for external devices. The layer offers a standardized API to access the peripheral devices of an ECU regardless of their location (i.e., whether the peripherals are internal or external with respect to the ECU's microcontroller) and their connection (e.g., via port pins, via a serial peripheral interface a.s.o.). The task of this layer is to make higher software layers independent of the ECU's hardware layout.

Services layer: This layer is mainly located on top of the ECU abstraction layer and provides operating system services, vehicle network communication and management services, memory services (e.g., nonvolatile random access memory management), diagnostic services (e.g., error logger), and state management services of the whole ECU. The task of this layer is the provision of basic services for the application software and other system software modules.

Complex device drivers: The concept of complex device drivers somehow intentionally violates the AUTOSAR-layered architecture to provide the possibility to deploy legacy device drivers in an AUTOSAR system software module stack. Additionally, the complex device driver module concept has been introduced to facilitate the integration of highly application and ECU-dependent drivers that require complex sensor evaluation and actuator control with direct access to the microcontroller itself and complex microcontroller peripherals for performance reasons.

Runtime Environment (`Rte`): The AUTOSAR Runtime Environment provides the interface between application software components and the basic software modules as well as the infrastructure services that enable communication between application software components.

Application layer: Actually this layer is not part of the AUTOSAR system software modules layered architecture, as this layer contains the AUTOSAR application software components described in Section 18.7.1.

In addition to this, mainly vertical structuring AUTOSAR further horizontally subdivides the system software modules into different substacks. This subdivision is depicted in Figure 18.13.

I/O substack: The I/O substack comprises software modules that provide standardized access to sensors, actuators, and ECU onboard peripherals (e.g., D/A or A/D converters, etc.).

Memory substack: The memory substack comprises software modules that facilitate the standardized access to internal and external nonvolatile memory for means of persistent data storage.

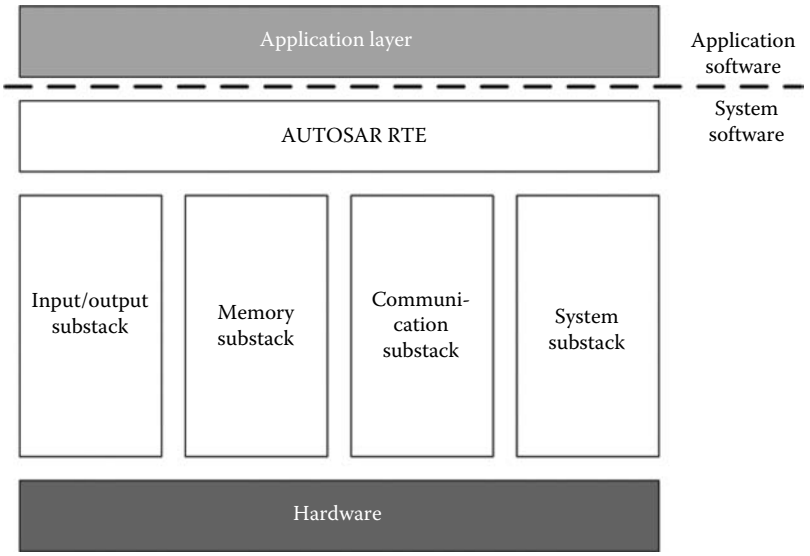


FIGURE 18.13 AUTOSAR—system software substacks overview.

Communication substack: The communication substack contains software modules that provide standardized access to vehicle networks (i.e., the local interconnect network (LIN) [8], the CAN [23,24], and FlexRay [12,27]).

System service substack: Last but not least, the system service substack encompasses all software modules that provide standardized (e.g., operating system, timer support, error loggers) and ECU-specific (ECU state management, watchdog management) system services, and library functions.

Independent of the vertical and horizontal structuring, the following classification can be applied to the AUTOSAR system software modules:

Drivers: A driver contains the functionality to "control and access an internal or an external device." Hereby, internal devices, which are located within the microcontroller, are controlled by internal drivers, whereas external devices, which are located on ECU hardware outside the microcontroller, are controlled by external drivers. Internal drivers can usually be found within the microcontroller abstraction layer, whereas external drivers are found in the ECU abstraction layer. Drivers do not change the content of the data handed to them.

Interfaces: An interface contains the functionality to "abstract from the hardware realization of a specific device" and to provide a generic API to access a specific type of device independent of the number of existing devices of that type and independent of the hardware realization of the different devices. Interfaces are generally located within the ECU abstraction layer. Interfaces do not change the content of the data handed to them.

Handlers: A handler "controls the concurrent, multiple, and asynchronous accesses of one or more clients" to one or more driver or interface modules. Thus, a handler performs buffering, queuing, arbitration, and multiplexing. Handlers do not change the content of the data handed to them.

Managers: A manager offers specific services for multiple clients. Managers are required whenever pure handler functionality is insufficient for accessing and using interface and driver modules. Managers furthermore are allowed to *evaluate and change or adapt the content of the data* handed to them. Managers are usually located in the services layer.

In the following sections the different substacks of AUTOSAR system software modules are described in detail.

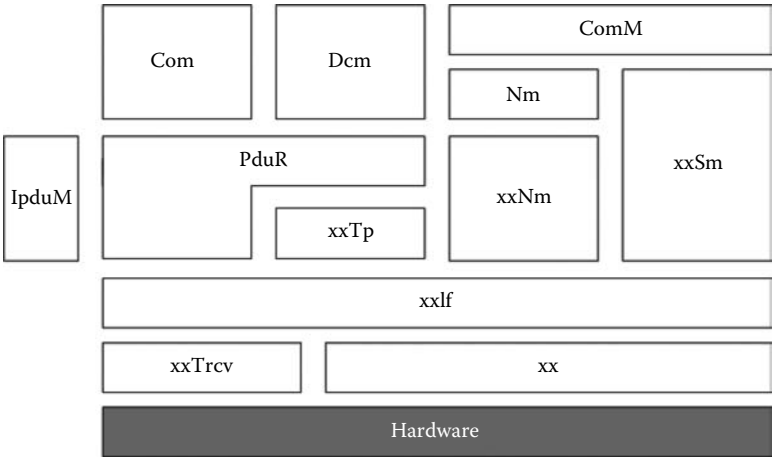


FIGURE 18.14 AUTOSAR—communication substack.

18.7.2.1 Communication Substack

The communication substack contains a group of modules that facilitate communication among the different ECUs in a vehicle via automotive communication protocols (CAN, LIN, and FlexRay). The structure of the communication substack is depicted in Figure 18.14.

Hereby `xx` is used as a placeholder for the respective communication protocol (i.e., CAN, LIN, FlexRay). Thus the AUTOSAR communication substack contains communication protocol specific instances of the Transport Protocol (`Tp`), Network Management (`Nm`), Interface (`If`), State Manager (`Sm`), Transceiver Driver (`Trcv`), and Driver (no suffix) modules. In the following, the different modules of the communication substack are described in detail.

Driver (*`xx`*): The Driver module (`Fr`, `Can`, `Lin`) provides the basis for the respective Interface module, by facilitating the "transmission and the reception of frames" via the respective CC. Hereby, the Driver is designed to handle multiple CCs of the same type. Thus, if an ECU contains, for example, FlexRay CCs of two different types, two different FlexRay Driver modules are required.

Transceiver driver (*`xxTrcv`*): The various different Transceiver Driver modules (`FrTrcv`, `CanTrcv`, `LinTrcv`) provide API functions for "controlling the transceiver hardware" (i.e., switching the transceivers into special modes [e.g., listen only mode]) and for obtaining diagnostic information from the transceiver hardware (e.g., information about short circuits of the different bus-lines of CAN or information about wake-up events on the bus).

Interface (*`xxIf`*): Using the frame-based services provided by the Driver module, the Interface module (`FrIf`, `CanIf`, `LinIf`) facilitates "the sending and the reception of protocol data units (PDUs)." Hereby, multiple PDUs can be packed into one frame at the sending ECU and have to be extracted again at the receiving ECU.* The point in time when this packing and extracting of PDUs takes place is governed by the temporal scheduling of so-called "communication jobs" of the FlexRay and the LIN Interface. The instant when the frames containing the packed PDUs are handed over to the Driver module for transmission or retrieved from the Driver module upon reception is triggered by communication jobs of the Interface module as well. In FlexRay, the schedule of these

* Currently only the FlexRay Interface modules support the packing of multiple PDUs into one frame. For the CAN and the LIN interface modules, there is a 1:1 relationship between PDUs and frames.

communication jobs is aligned with the communication schedule, in LIN the schedule of the LIN Interface module governs the communication schedule on the LIN bus. In contrast to this in CAN, the temporal schedule of the PDU transmission is governed by the `Com` module (see below). In FlexRay, each communication job can consist of one or more "communication operations," each of these communication operations handling exactly one communication frame (including the PDUs contained in this frame).

The interface modules are designed to be able to deal with multiple different drivers for different types of CCs (e.g., freescale's MFR4300 or FlexRay CCs based on the BOSCH E-Ray core in the FlexRay case). Furthermore, the Interface module wraps the API provided by the Transceiver Driver module and provides support for multiple different Transceiver Driver modules (similar to the support for multiple different Driver modules).

Transport protocol (`xxTp`): The transport protocol is used to perform "segmentation and reassembly of large PDUs." On CAN (`CanTp`), this protocol is compatible (in certain configuration settings) to ISO TP (see Section 18.5.1). Just like ISO TP, the user of the services provided by the transport protocol is the diagnostic layer, called Diagnostic Communication Manager in AUTOSAR (see below).

Network management (`xxNm`, `Nm`): Similar to OSEK NM (see Section 18.3.2), the AUTOSAR NM modules provide means for the coordinated transition of the ECUs in a network into and out of a low-power (or even power down) sleep mode. AUTOSAR NM is hereby divided into two modules: a communication protocol-independent module named "*Generic NM* (`Nm`)" and communication protocol-dependent modules named *FlexRay NM* (`FrNm`) and *CAN NM* (`CanNm`).*

State manager (`xxSm`): The State Manager modules (`CanSm`, `LinSm`, and `FrSm`) facilitate the "state management of the respective CC" with respect to communication system-dependent start-up and shutdown features and provide a common state machine API to the upper layer (i.e., the Communication Manager [`ComM`]). This API consists of functions for requesting the communication modes `Full`, `Silent` (i.e., listen only), and `No Communication`.

PDU router (`PduR`): The PDU Router module provides two major services. On the one hand, it "dispatches PDUs received via the underlying modules" (i.e., Interface and Transport Layer modules) to the different higher layers (`Com`, `Dcm`). On the other hand, the PDU router "performs gateway functionalities" between different communication networks by forwarding PDUs from one interface to another of either the same (e.g., FlexRay to FlexRay) or of different type (e.g., CAN to FlexRay). Routing decisions in the PDU Router are based on a static routing table and on the identifiers of the PDUs.

PDU multiplexer (`IpduM`): The PDU Multiplexer module takes care of "multiplexing parts of a PDU." Hereby, the value of a dedicated part of the PDU (the "multiplexer switch") is used to define the semantic content of the remainder of the PDU (just like the tag element in a variant record or a union in programming languages). In the reception case, multiplexed PDUs are forwarded from the `PduR` to the `IpduM` for demultiplexing. Once demultiplexed, the `IpduM` hands the PDUs back to the `PduR`. In the sending case, the `PduR` obtains a PDU from `Com` and hands this PDU to the `IpduM` for multiplexing. The `IpduM` returns the multiplexed PDU to the `PduR`, who routes the multiplexed PDU to its final destination.

Communication (`Com`): The `Com` module provides "signal-based communication" to the upper layer (`Rte`). The signal-based communication service of `Com` can be used for intra-ECU communication as well as for inter-ECU communication. In the former case, `Com` mainly uses shared memory for this

* As LIN is a master–slave bus, no `LinNm` is required, as the LIN master dictates when the network shall transit into a low-power mode.

intra-ECU communication, whereas for the latter case at the sender side `Com` packs multiple signals into a PDU and forwards this PDU to the `PduR` to issue the PDU's transmission via the respective Interface module. On the receiver side, `Com` obtains a PDU from the PDU router, extracts the signals contained in the PDU and forwards the extracted signals to the upper software layer (`Rte`).

Diagnostic Communication Manager (`Dcm`): The Diagnostic Communication Manager module is a submodule of the AUTOSAR diagnostic module. The `Dcm` module provides "services which allow a tester device to control diagnostic functions" in an ECU via the communication network (i.e., CAN, LIN, FlexRay). Hereby, the `Dcm` supports the diagnostic protocols KWP2000 and UDS (see Section 18.5.2 for details).

Communication Manager (`ComM`): The Communication Manager is a resource manager which "encapsulates the control of the underlying communication services." The `ComM` collects bus communication access requests from communication requesters (e.g., `Dcm`) and coordinates these requests by interacting with `Nm` and the respective State Manager modules. This way the `ComM` provides a simplified API to the `Nm`, where a user of the API does not require any knowledge of the particular communication channel to use. Via the `ComM` API, a user simply request a specific communication mode (i.e., `Full`, `Silent`, or `No Communication`) and the `ComM` switches (based on a statically configured table mapping users to channels) the communication capability of the corresponding channel(s) to `On`, `Silent`, or `Off`.

18.7.2.2 Memory Substack

The memory substack contains a group of modules that facilitate handling of the ECU's onboard nonvolatile memory (i.e., providing API functions to store and retrieved data in the ECU's nonvolatile memory (e.g., flash EEPROM or EEPROM). The structure of the memory substack is depicted in Figure 18.15.

In the following, the different modules of the memory substack are described in detail.

Flash driver (`Fls`): The Flash Driver provides services for "reading from, writing to, and erasing parts of the ECU's flash memory." Furthermore the Flash Driver facilitates the setting and resetting of the write/erase protection of the flash EPPROM if such protection is supported by the underlying hardware.

EEPROM driver (`Eep`): The EEPROM Driver provides services for reading from, writing to, and erasing parts of the ECU's EEPROM. In addition to these basic services, the EEPROM Driver

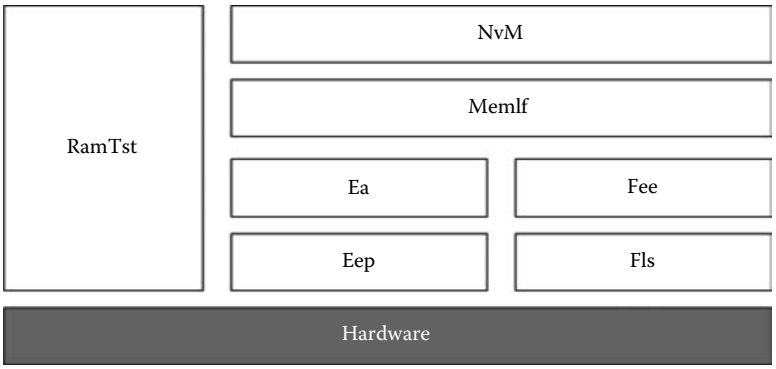


FIGURE 18.15 AUTOSAR—memory substack.

provides a service for comparing a data block in the EEPROM with a data block in the memory (e.g., RAM).

Flash EEPROM emulation (`Fee`): The Flash EEPROM Emulation module "emulates EEPROM functionality" using the services provided by Flash Driver module. By making use of multiple flash sectors and smart copying of the data between these sectors, the `Fee` simulates an EEPROM-like behavior, i.e., the possibility to perform program/erase operations on subsector granularity.

EEPROM abstraction (`Ea`): The EEPROM Abstraction module "provides uniform mechanisms to access the ECU's internal and external EEPROM devices." It abstracts from the location of peripheral EEPROM devices (including their connection to the microcontroller), the ECU hardware layout, and the number of EEPROM devices.

Memory abstraction interface (`MemIf`): The Memory Abstraction Interface module allows the Nonvolatile RAM Manager module (see below) to "access several memory abstraction modules (`Fee` or `Ea` modules) in a uniform way." Hereby, the `MemIf` abstracts from the number of underlying `Fee` or `Ea` modules providing a runtime translation of each block access initiated by the Nonvolatile RAM manager module to select the corresponding driver functions which are unique for all configured EEPROM or flash EEPROM storage devices.

Nonvolatile RAM manager (`NvM`): The Nonvolatile RAM manager module provides services to "ensure the data storage and maintenance of nonvolatile data" according to their individual requirements in an automotive environment, namely, synchronous as well as asynchronous services for the initialization, the reading, the writing, and the control of nonvolatile data. The `NvM` operates on a block basis, where the following types of blocks are distinguished: For "native blocks" the `NvM` provides a "RAM mirror," which contains a copy of the data stored in the nonvolatile memory block. This RAM mirror is initialized with the data from the nonvolatile block upon ECU power-up. Upon ECU shutdown, the data from the RAM mirror is flushed to the corresponding nonvolatile memory block. Additionally, the `NvM` provides API services which can force the transfer of a memory block from nonvolatile memory into the corresponding RAM mirror and vice versa. In addition to the facilities of native blocks "redundant blocks" provide enhanced fault tolerance, reliability, and availability. Due to replication of the redundant block in nonvolatile memory, the resilience against data corruption is increased.

RAM test (`RamTst`): The RAM Test module performs "functional tests of the ECU's internal RAM cells." "Complete" tests are performed upon ECU start-up and shutdown as well as on request by special diagnostic commands. During operation "partial test" are performed in a periodic manner (e.g., block by block or cell by cell). For both types of tests several RAM test algorithms, which have been chosen according to the IEC 61508 standard, are available. Depending on the algorithms' diagnostic coverage rate, the algorithms are divided into the following categories: Group 1 (low) with a diagnostic coverage rate smaller than 60%, group 2 (medium) exhibiting a diagnostic coverage rate of 60%–90%, and group 3 (high) with a diagnostic coverage rate of 90%–99%.

18.7.2.3 I/O Substack

The I/O substack contains a group of modules that facilitate handling of the ECU's I/O capabilities. The structure of the I/O substack is depicted in Figure 18.16.

In the following, the different modules of the I/O substack are described in detail.

Interrupt capture unit driver (`Icu`): The interrupt capture unit (ICU) Driver is a "module using the ICU hardware" to implement services like signal edge notification, controlling of wake-up interrupts, periodic signal time measurement, edge time stamping (usable for the acquisition of non-periodic signals), and edge counting.

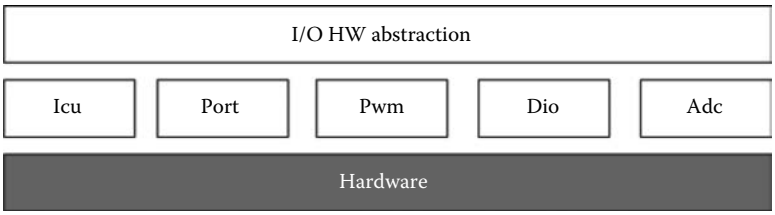


FIGURE 18.16 AUTOSAR—I/O substack.

Port driver (`Port`): The Port Driver module provides the service for "initializing the whole port structure" of the microcontroller, allowing for the configuration of different functionalities for each port and port pin (e.g., analog digital conversion (ADC), digital I/O (DIO), etc.). Hereby, the port pin direction (I/O), the initial level of the port pin, and the fact whether the port pin direction is modifiable during run-time are part of this configuration. Other I/O drivers (e.g., `Dio`, `Adc`, etc.) rely on the configuration performed by the Port Driver.

Pulse width modulation (`Pwm`) *driver*: The PWM Driver module provides functions to "initialize and control the hardware PMW unit" of the microcontroller. The `Pwm` module thus allows for the generation of pulses with variable pulse width by facilitating the selection of the duty cycle and the signal period time. The `Pwm` module supports multiple PWM channels, where each channel is linked to a hardware PWM unit which belongs to the microcontroller.

Digital I/O driver (`Dio`): The DIO Driver provides services for "reading from and writing to DIO channels" (i.e., port pins), DIO ports, and groups of DIO channels. Hereby, the `Dio` modules work on pins and ports which have been properly configured by the Port Driver for this purpose.

Analog/digital converter (`Adc`) *driver*: The ADC Driver module "initializes and controls the internal ADC unit(s)" of the microcontroller. The module provides services to start and stop an analog to digital conversion respectively to enable and disable the trigger source for a conversion. Furthermore, the module provides services to enable and disable a notification mechanism and routines to query status and result of a conversion. The `Adc` module works on so-called ADC channel groups. An ADC channel group combines an ADC channel (i.e., an analog input pin), the needed ADC circuitry itself, and a conversion result register into an entity that can be individually controlled and accessed via the `Adc` module.

I/O hardware abstraction: The I/O Hardware Abstraction module provides a signal-based interface to internal and external I/O devices of an ECU. Hereby, the module abstracts from whether a certain I/O device is an MCU internal device, or whether a device is externally connected to the MCU, by performing static normalization/inversion of values according to their physical representation at the inputs/outputs of the ECU hardware (i.e., static influences, like voltage division or hardware inversion, on the path between the I/O device and the MCU port pin are compensated).

18.7.2.4 System Services Substack

The system services substack contains a group of modules that can be used by modules of all AUTOSAR layers. Examples are real-time-operating system, error handler, and library functions (like CRC and interpolation functions). The structure of the system services substack (excluding the library functions) is depicted in Figure 18.17.

In the following the different modules of the system services substack are described in detail.

Operating system (`Os`): The AUTOSAR-operating system provides "real-time-operating system services" to both the other system software modules and to the application software components of

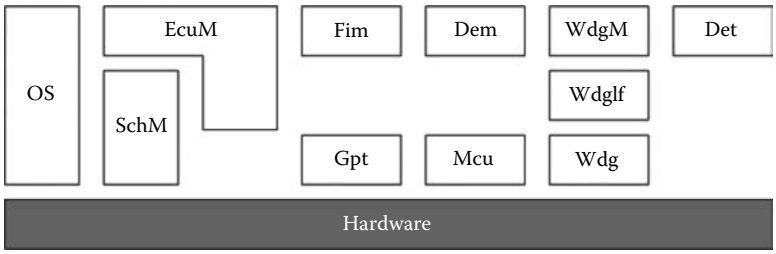


FIGURE 18.17 AUTOSAR—system services substack.

AUTOSAR. The `Os` module is configured and scaled statically, provides a priority-based scheduling policy and protective functions with respect to memory and timing at run-time, and is designed to be hostable on low-end controllers.

Similar to the OSEKtime dispatcher tables (see Section 18.3.4), AUTOSAR `Os` provides so-called "schedule tables" consisting of one or more "expiry points." Hereby, each expiry point is assigned an offset measured in `Os` ticks from the start of the schedule table. Once an expiry point is reached, the action corresponding to the expiry point (e.g., the activation of a task or the setting of an event) is processed. At runtime, the `Os` iterates over the schedule table, processing each expiry point in turn. The iteration is driven by an `Os` counter. To facilitate the execution of tasks synchronous to external events (e.g., synchronous to the FlexRay communication schedule), schedule tables can be synchronized to external time sources (e.g., FlexRay's global time).

As far as protection against timing violations is concerned, AUTOSAR `Os` does not provide deadline monitoring (as does OSEKtime OS), but provides the facility to track the execution time of each task and ISR and to raise an error in case either exceeds its statically assigned execution time budgets. Regarding memory protection, AUTOSAR `Os` uses (similar to the protected OSEK defined by HIS [see Section 18.4.1]) the memory protection unit of the MCU to provide coarse-grained memory protection of the different tasks against each other.

Basic software (BSW) *scheduler* (`SchM`): The BSW Scheduler module provides means to embed other AUTOSAR system software module implementations into the context of an AUTOSAR `Os` task or ISR, triggers main processing functions of the system software modules, and applies data consistency mechanisms for these modules. Just like the `Rte` provides the infrastructure for software components by embedding runnable entities in a task context, the `SchM` module "provides the infrastructure for the other system software modules" by embedding their main processing functions in a task context.*

ECU State Manager (`EcuM`): The ECU State Manager module manages all aspects of the ECU related to the `Off`, `Run`, and `Sleep` states of that ECU and the transitions between these states like "start-up and shutdown." In detail, the ECU State Manager is responsible for the initialization and de-initialization of all basic software modules including `Os` and `Rte`, cooperates with the `ComM`, and hence indirectly with `Nm`, to shut down the ECU when needed, manages all wake-up events, and configures the ECU for `Sleep` state when requested. To fulfill all these tasks, the EcuM provides some important protocols: the "run request protocol," which is needed to coordinate whether the ECU must be kept alive or is ready to shut down, the "wakeup validation protocol" to distinguish "real" wakeup events from "erratic" ones, and the "time-triggered increased inoperation protocol," which allows to put the ECU into an increasingly energy saving sleep state.

* Usually the main processing functions of multiple system software modules are embedded into a single task to keep the number of tasks required for execution of the whole AUTOSAR system software low.

Function inhibition manager (`Fim`): The Function Inhibition Manager is a submodule of the AUTOSAR diagnostic module. The `Fim` is responsible for providing an "execution control mechanism for the runnables" of application software components and system software modules. By means of the `Fim`, these runnables can be inhibited (i.e., deactivated) according to the `Fim`'s static configuration. The functionalities of the runnables are assigned to a unique function identifier (FID) along with an inhibit conditions for that particular FID. The functionalities poll for the permission state of their respective FIDs before execution. If an inhibit condition is true for a particular FID, the corresponding functionality is not executed anymore.

The `Fim` is closely related to the `Dem` as diagnostic events and their status information can serve as possible inhibit conditions. Hence, functionality which needs to be stopped in case of a failure can be represented by a particular FID. If the failure is detected and the event is reported to the `Dem`, the `Fim` then inhibits the FID and therefore the corresponding functionality.

Diagnostic event manager (`Dem`): Like the `Dcm` module and the `Fim` module, the Diagnostic Event Manager module is a submodule of the diagnostic module within AUTOSAR. The `Dem` is responsible for "processing and persistently storing diagnostic events/errors"* and associated data (so-called freeze frame data). To facilitate the persistent storage of these diagnostic trouble codes (DTCs), the `Dem` makes use of the services provided by the `NvM`. Application software components as well as other system software modules can raise diagnostic event by means of `Dem` API calls.

The diagnostic events registered by the `Dem` serve as triggers for state updates of the `Fim` and thus might lead to the inhibition of certain runnables. Upon request of the `Dcm`, the `Dem` provides an up-to-date list of the currently stored DTCs, which are then sent to a tester client by means of the `Dcm` services.

Watchdog driver (`Wdg`): This module provides services for initialization, changing of the operation mode (`Fast`, `Slow`, `Off`) and "triggering the ECU's watchdog device." In case an ECU provides multiple different watchdog devices (e.g., internal and external devices), a dedicated `Wdg` module has to be present for each of the devices.

Watchdog interface (`WdgIf`): In case of more than one watchdog device and corresponding Watchdog Driver (e.g., both an internal software watchdog and an external hardware watchdog) is being used on an ECU, the Watchdog Interface module allows the Watchdog Manager module (see below) to select the correct Watchdog Driver—and thus the watchdog device—via a device index, while retaining the API and functionality of the underlying driver. Thus, the `WdgIf` module provides "uniform access to services of the underlying Watchdog Drivers" like mode switching and triggering.

Watchdog manager (`WdgM`): The Watchdog Manager module is intended to "supervise the (periodic) execution" of application software components or other system software modules. Via the services provided by the `WdgIf`, the `WdgM` abstracts from the triggering of hardware watchdog entities and itself provides supervision capabilities to an (theoretically) infinite number of clients. The `WdgM` provides an individual port for each supervised client, where the clients have to indicate their proof of aliveness by updating an alive-counter. Within a cyclic scheduled main processing function of the `WdgM`, the alive-counters of all supervised clients are checked against their own independent timing constraints and a "global supervision status" is derived. Based on this global supervision status, the `WdgM` decides whether or not to trigger the hardware watchdog via the `WdgIf`'s API. Hereby, the set of supervised clients and their individual timing constraints is defined by configurable parameters of the `WdgM`.

Development error tracer (`Det`): The Development Error Tracer module is the central instance where all other system software modules "report detected development errors" to. The API parameters

* UDS and KWP2000 use the term DTC for these events.

handed to the `Det` allow for tracing source and kind of error, namely, the module and the function in which error has been detected and the type of the error. The functionality behind the API of the `Det` is not specified in AUTOSAR. Possible functionalities could be the setting of debugger breakpoints within the error reporting API, the counting of the number of reported errors, the logging of `Det` calls together with the passed parameters to a RAM buffer for later analysis, and the sending of reported errors via some communication interface (e.g., `CanIf`) to external logger devices.

General purpose timer (`Gpt`) *driver*: The general purpose timer (GPT) driver module provides services for starting and stopping a functional timer instance within the hardware general-purpose timer module and thus "provides exact and short-term timings" for use in the `Os` or within other system software modules where an `Os` alarm service causes too much overhead. Individual timeout periods (single shot mode) as well as repeating timeout periods (continuous mode) can be generated via the `Gpt` module. The user can configure whether a notification shall be invoked when the requested timeout period has expired. These notifications can be enabled and disabled at runtime. Both the relative time elapsed since the last notification occurred and the time remaining until the next notification will occur can be queried via API functions of the `Gpt` module.

Microcontroller unit (`Mcu`) *driver*: The MCU Driver module provides services for "basic microcontroller initialization, power-down functionality, microcontroller reset" and microcontroller-specific functions required from other system software modules. The initialization services of the MCU Driver module allow a flexible and application-related MCU initialization in addition to the start-up code.* The services of the MCU Driver include the initialization of the MCU's clock, initialization of the MCU's phase-locked loop, the initialization of clock prescalers, and the configuration of the MCU's clock distribution. Furthermore, the MCU Driver takes care of the initialization of the MCU's RAM sections, facilitates the activation of the MCU's reduced power modes (i.e., putting the MCU into a low-power mode), and provides a service for enforcing a reset of the MCU and a service for obtaining the reset reason from the MCU hardware.

18.8 JasPar

Founded September 2004 by Toyota, Nissan, Honda, and Toyotsu Electronics, the nonprofit organization JasPar [4] was created to develop a unified architecture for vehicle networking and software development tailored to the requirements of Japanese customers.

JasPar's goals are focused on the evaluation, the exploitation, and the implementation of existing standards in actual projects. The organization seeks to serve as the Japanese point of contact for automotive-standards organizations in other regions. Based on the results of the AUTOSAR and FlexRay consortium, JasPar was constructed as complementary organization to unite standardization and product attempts in Japan (see Figure 18.18) to maintain the possibility of tailored product generation for the Japanese OEM and finally the end customer.

Within JasPar, working groups are focusing on developing design tools and prototypes for automotive networks, creating standards for in-vehicle cable routing, writing guidelines for wire-harness design, developing recommended circuits for controllers/transceivers, and planning communication software. Moreover, JasPar is feeding suggestions back to related consortia for new standard requirements.

An example given in the evaluation of existing standards is the standardization of the business process in using FlexRay parameter settings. This standardization shall reduce the risk of failures

* The start-up code itself is not within the scope of AUTOSAR.

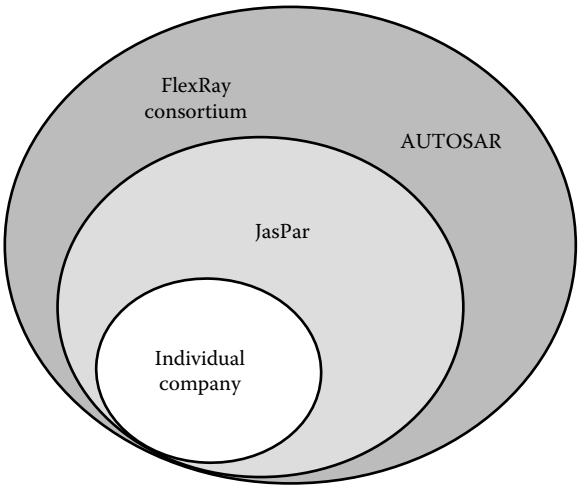


FIGURE 18.18 Relationship AUTOSAR, FlexRay, JasPar consortium.

caused by improper parameter settings and thus shall prevent a reduction of overall quality, which is a crucial product criteria on the Japanese market. Further activities of JasPar include interoperability tests between different FlexRay CCs and the evaluation of bus wiring conditions including the observation of the communication behavior of FlexRay at different (currently unspecified) transmission rates.

18.9 Summary

Since 1993, the major automotive companies have been striving for the deployment of standard software modules in their applications to achieve an increased test coverage and higher reliability requirements that can only be met if standardized modules are used at various system levels.

This chapter provided an overview of today's industry practices in standardized automotive system software. Existing standards proposed by industry partnerships like OSEK/VDX, HIS, JasPar, and AUTOSAR, and by standardization authorities like ISO have been presented. Of all presented approaches, the AUTOSAR partnership which started off in May 2003 and aims at putting software modules according to the AUTOSAR standard into production vehicles by 2008 (BMW and Honda) [14] turns out to be the most promising one.

This is on the one hand due to the fact that several of already approved standards used today (like OSEK OS and COM, HIS CAN driver, and ISO transport layer and diagnostics) heavily inspired the corresponding AUTOSAR standards, ensuring that the AUTOSAR standard is built on solid well-proven technology. On the other hand the AUTOSAR open standard has massive industrial backup: All AUTOSAR core members including seven of the world's biggest vehicle manufacturers who together account for about 55% of all vehicles produced are strongly committed to the project and plan to release tested and verified AUTOSAR 4.0 specifications together with the corresponding conformance test specifications by September 2009 [1].

References

1. AUTOSAR—Top Level Schedule Phase II. Consortium Web Page.
2. EASIS—Electronic Architecture and System Engineering for Integrated Safety Systems. Project Web Page.
3. HIS—Herstellerinitiative Software. Project Web Page.

4. JASPAR—Japan Automotive Software Platform Architecture. Consortium Web Page.
5. OSEK/VDX. Project Web Page.
6. V. Barthelmann, A. Schedl, E. Dilger, T. Führer, B. Hedenetz, J. Ruh, M. Kühlewein, E. Fuchs, Y. Domaratsky, A. Krüger, P. Pelcat, M. Glück, S. Poledna, T. Ringler, B. Nash, and T. Curtis. OSEK/VDX—Time-triggered operating system, Version 1.0. Technical report, July 2001.
7. AUTOSAR Consortium. AUTOSAR—Layered software architecture. Version 2.2.1. Technical report, Release 3.0, Rev. 0001, AUTOSAR Consortium, February 2008.
8. LIN Consortium. LIN specification package. Version 2.1. Technical report, LIN Consortium, November 2006.
9. J. Spohr et al. OSEK/VDX—Communication, Version 3.0.3. Technical report, July 2004.
10. J. Spohr et al. OSEK/VDX—System generation—OIL: OSEK implementation language, Version 2.5. Technical report, July 2004.
11. H. Fennel, S. Bunzel, H. Heinecke, J. Bielefeld, S. Fürst, K. -P. Schnelle, W. Grote, N. Maldener, T. Weber, F. Wohlgemuth, J. Ruh, L. Lundh, T. Sandén, P. Heitkämper, R. Rimkus, J. Leflour, A. Gilberg, U. Virnich, S. Voget, K. Nishikawa, K. Kajio, K. Lange, T. Scharnhorst, and B. Kunkel. Achievements and exploitation of the AUTOSAR development partnership. In *Proceedings of the Convergence 2006*, Detroit, MI, October 2006.
12. T. Führer, F. Hartwich, R. Hugel, and H. Weiler. FlexRay—The communication system for future control systems in vehicles. In *Proceedings of the SAE 2003 World Congress and Exhibition*, Detroit, MI, March 2003. Society of Automotive Engineers.
13. J. Goodenough and L. Sha. The priority ceiling protocol: A method for minimizing the blocking of high-priority Ada tasks. Technical report SEI-SSR-4, Software Engineering Institute, Pittsburgh, PA, May 1988.
14. P. Hansen. AUTOSAR standard software architecture partnership takes shape. *The Hansen Report on Automotive Electronics*, 17(8):1–3, October 2004.
15. P. Hansen. New S-class Mercedes: Pioneering electronics. *The Hansen Report on Automotive Electronics*, 18(8):1–2, October 2005.
16. HIS (Hersteller Initiative Software). HIS functional specification of a flash driver, Version 1.3. Technical report, June 2002.
17. HIS (Hersteller Initiative Software). HIS CAN driver specification, Version 1.0. Technical report, August 2003.
18. HIS (Hersteller Initiative Software). OSEK OS extensions for protected applications, Version 1.0. Technical report, July 2003.
19. HIS (Hersteller Initiative Software). API IO driver, Version 2.1.3. Technical report, April 2004.
20. HIS (Hersteller Initiative Software). API IO library, Version 2.0.3. Technical report, March 2004.
21. C. Hoffmann, J. Minuth, J. Krammer, J. Graf, K. J. Neumann, F. Kaag, A. Maisch, W. Roche, O. Quelenis, E. Farges, P. Aberl, D. John, L. Mathieu, M. Schütze, D. Gronemann, and J. Spohr. OSEK/VDX—Network management—Concept and application programming interface, Version 2.5.3. Technical report, July 2004.
22. ISO. Road Vehicles—Diagnostic systems—Keyword protocol 2000—Part 3: Application layer. Technical report ISO/DIS 14230-3, ISO (International Organization for Standardization), Geneva, Switzerland, 1999.
23. ISO. Road Vehicles—Controller area network (CAN)—Part 1: Data link layer and physical signalling. Technical report ISO/DIS 11898-1, ISO (International Organization for Standardization), Geneva, Switzerland, 2003.
24. ISO. Road vehicles—Controller area network (CAN)—Part 2: High-speed medium access unit. Technical report ISO/DIS 11898-2, ISO (International Organization for Standardization), Geneva, Switzerland, 2003.

25. ISO. Road vehicles—Diagnostics on controller area networks (CAN)—Part 2: Network layer services. Technical report ISO/DIS 15765-2.2, ISO (International Organization for Standardization), Geneva, Switzerland, April 2003.
26. ISO. Road vehicles—Unified diagnostic services (UDS)—Part 1: Specification and requirements. Technical report ISO/DIS 14229-1, ISO (International Organization for Standardization), Geneva, Switzerland, 2004.
27. R. Mores, G. Hay, R. Belschner, J. Berwanger, C. Ebner, S. Fluhrer, E. Fuchs, B. Hedenetz, W. Kuffner, A. Krüger, P. Lohrmann, D. Millinger, M. Peller, J. Ruh, A. Schedl, and M. Sprachmann. FlexRay—The communication system for advanced automotive control systems. In *Proceedings of the SAE 2001 World Congress*, Detroit, MI, USA, March 2001. Society of Automotive Engineers.
28. A. Schedl, E. Dilger, T. Führer, B. Hedenetz, J. Ruh, M. Kühlewein, E. Fuchs, T. M. Galla, Y. Domaratsky, A. Krüger, P. Pelcat, M. Taï-Leung, M. Glück, S. Poledna, T. Ringler, B. Nash, and T. Curtis. OSEK/VDX—Fault-tolerant communication, Version 1.0. Technical report, July 2001.
29. R. Schuermans, R. Zaiser, F. Hepperle, H. Schröter, R. Motz, A. Aberfeld, H.-G. Kunz, T. Tyl, R. Leinfellner, H. Amsbeck, H. Styrsky, B. Ruoff, and L. Wahlmann. XCP—The universal measurement and calibration protocol family, Version 1.0. Technical report, Association for Standardisation of Automation and Measuring Systems (ASAM), April 2003.
30. T. Wollstadt, W. Kremer, J. Spohr, S. Steinhauer, T. Thurner, K. J. Neumann, H. Kuder, F. Mosnier, D. Schäfer-Siebert, J. Schiemann, R. John, S. Parisi, A. Zahir, J. Söderberg, P. Mortara, B. France, K. Suganuma, S. Poledna, G. Göser, G. Weil, A. Calvy, K. Westerholz, J. Meyer, A. Maisch, M. Geischeder, K. Gresser, A. Jankowiak, M. Schwab, E. Svenske, M. Tchervinsky, K. Tindell, G. Göser, C. Thierer, W. Janz, and V. Barthelmann. OSEK/VDX—Operating system, Version 2.2.3. Technical report, February 2005.

19

Volcano: Enabling Correctness by Design

19.1 Introduction 19-2
19.2 Volcano Concepts 19-4
19.3 Volcano Signals and the Publish/Subscribe Model 19-5
19.4 Update Bits 19-5
19.5 Flags 19-6
19.6 Timeouts 19-6
19.7 Frames 19-6
Immediate Frames • Frame Modes
19.8 Network Interfaces 19-7
19.9 Volcano API 19-7
Volcano Thread-of-Control
19.10 Volcano Resource Information 19-8
19.11 Execution Time of Volcano Processing Calls 19-8
19.12 Timing Model 19-8
19.13 Jitter 19-9
19.14 Capture of Timing Constraints 19-10
19.15 Volcano Network Architect 19-11
Car OEM Tool Chain: One Example
19.16 VNA: Tool Overview 19-13
Global Objects • Project or Configuration Related Data (Projects, Configurations, Releases • Database • Version and Variant Handling • Consistency Checking • Timing Analysis/Frame Compilation • Volcano Filtering Algorithm • Multiprotocol Support • Gatewaying • Data Export and Import
19.17 Volcano Configuration 19-17
19.18 Configuration Files 19-17
Fixed Information • Private Information • Network Information • Target Information
19.19 Workflow 19-18
Acknowledgment 19-20
Reference 19-20
Bibliography 19-20
More Information 19-20

Antal Rajnak
Mentor Graphics Corporation

19.1 Introduction

Volcano is a holistic concept defining a protocol-independent design methodology for distributed real-time networks in vehicles. The concept is dealing with both technical and nontechnical entities (i.e., partitioning of responsibilities into well-defined roles in the development process).

The vision of Volcano is "Enabling Correctness by Design." By taking a strict systems engineering approach and focusing resources into design, a majority of system-related issues can be identified and solved early in a project. The quality is designed into the vehicle, not tested out. Minimized cost, increased quality, and high degree of configuration/reconfiguration flexibility are the trademarks of the Volcano concept.

The Volcano approach is particularly beneficial as the complexity of vehicles is increasing very rapidly and as projects will have to cope with new functions and requirements throughout their lifetime.

A unique feature of the Volcano concept—the solution called "Post-Compile-Time Reconfiguration Flexibility" where the network configuration containing signal-to-frame mapping, ID assignment, and frame period—is located in a configurable flash area of the electronic control unit (ECU), and can be changed without the need for touching the application software, thus eliminating the need for revalidation, saving cost and lead time.

The concepts origins can be traced back to a project at Volvo Car Corporation in 1994–1998 when development of Volvo's new "Large Platform" has taken place (Figure 19.1). It is reusing solid industrial experience, and is taking into account recent findings from real-time research.

The concept is characterized by three important features:

- Ability to guarantee the real-time performance of the network already at the design stage, thus significantly reducing the need for testing
- Built-in flexibility enabling the vehicle manufacturer to upgrade the network in the preproduction phase of a project as well as in the aftermarket
- Efficient use of available resources

The actual implementation of the concept consists of two major parts:

- *Off-line tool set*: For requirement capturing and automated network design (covering multiple protocols and gateway configuration). It is providing strong administrative functions for variant and version handling, needed during the complete life cycle of a car project.

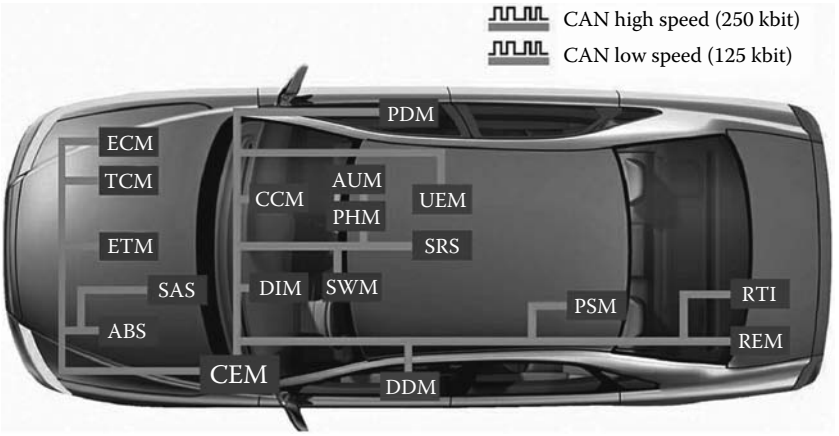


FIGURE 19.1 Volvo S80 main networks.

- *Target part*: Represented by a highly efficient and portable-embedded software package offering a signal-based application programmer interface (API), handling of multiple protocols, integrated gateway functionality, and post-compile-time reconfiguration capability, together with a PC-based generation tool.

The Volcano approach is particularly beneficial as the complexity of vehicles is increasing very rapidly and as projects will have to cope with new functions and requirements throughout their lifetime. The computing industry has discovered over the last 40 years that certain techniques are needed in order to manage complex software systems. Two of these techniques are abstraction (where unnecessary information is hidden) and composability (if software components proven to be correct are combined, then the resulting system will be correct as well). Volcano is making heavy use of both of these techniques.

The automotive industry is implementing an increasing number of functions in software. Introduction of protocols like MOST for multimedia and FlexRay for active chassis systems result in highly complex electrical architectures. Finally, all these complex subnetworks are linked through gateways. The behavior of the entire car network has a crucial influence upon the car's performance and reliability. To manage software development involving many suppliers, hundreds of thousands of lines of code and thousands of signals require a structured systems engineering approach. Inherent in the concept of systems engineering is a clear partitioning of the architecture, requirements, and responsibilities.

A modern vehicle includes a number of microprocessor-based components called ECUs, provided by a variety of suppliers.

Control area network (CAN) provides an industry-standard solution for connecting ECUs together using a single broadcast bus. A shared broadcast bus makes it much easier to add desired functionality: ECUs can be added easily, and they can communicate data easily and cheaply (adding a function may be "just software"). But increased functionality leads to more software and greater complexity. Testing a module for conformance to timing requirements is the most difficult of the problems. With a shared broadcast bus, the timing performance of the bus might not be known until all the modules are delivered and the bus usage of each is known. Testing for timing conformance can only then begin (which is often too far into the development of a vehicle to find and correct major timing errors). The supplier of a module can only do limited testing for timing conformance—they do not have a complete picture of the final load placed on the bus. This is particularly important when dealing with the CAN bus—arrivals of frames from the bus may cause interrupts on a module wishing to receive the frames, and so the load on the microprocessor in the ECU is partially dependent on the bus load.

It is often thought that CAN is somehow unpredictable and the latencies for lower priority frames in the network are unbounded. This is untrue, and in fact CAN is a highly predictable communications protocol. Furthermore, CAN is well suited to handle large amounts of traffic with differing time constraints.

However, with CAN there are a few particular problems:

- Distribution of identifiers
- CAN uses identifiers for two purposes: distinguishing different messages on the bus, and assigning relative priorities to those messages—the latter being often neglected
- Limited bandwidth, due to low maximum signaling speed of 1 Mbps, is further reduced by significant protocol overhead

Volcano was designed to provide abstraction, composability, and identifier distribution reflecting true urgencies, and at the same time providing the most efficient utilization of the protocol.

19.2 Volcano Concepts

The Volcano concept is founded on the ability to guarantee the worst-case latencies of all frames sent in a multiprotocol network system. This is a key step because it gives the following:

- A way of guaranteeing that there are no communications-related timing problems.
- A way of maximizing the amount of information carried on the bus. The latter is important for reduced production costs.
- The possibility to develop highly automated tools for the design of optimal network configurations.

The timing guarantee for CAN is provided by mathematical analysis developed from academic research [1]. Other protocols like FlexRay are predictable by design. For this reason, some of the subjects discussed below are CAN specific; others are independent of the protocol used.

The analysis is able to calculate the worst-case latency for each frame sent on the bus. This latency is the longest time from placing a frame in a CAN controller at the sending side to the time the frame is correctly received at all receivers.

The analysis needs to make several assumptions about how the bus is used.

One of these assumptions is that there is a limited set of frames that can access the bus, and that time-related attributes of these frames are known (e.g., frame size, frame periodicity, queuing jitter, and so on).

Another important assumption is that the CAN hardware can be driven correctly:

- Internal message queue within any CAN controller in the system is organized (or can be used) as such that the highest priority message will be sent out first if more than one message is ready to be sent. (The hardware-slot position-based arbitration is OK as long as the number of sent frames is less than the number of transmit-slots available in the CAN controller.)
- CAN controller should be able to send out a stream of scheduled messages without releasing the bus in the interframe space between two messages. Such devices will arbitrate for the bus right after sending the previous message and will only release the bus in case of lost arbitration.

A third important assumption is the "error model": the analysis can account for retransmissions due to errors on the bus, but requires a model for the number of errors in a given time interval.

The Volcano software running in each ECU controls the CAN hardware and accesses the bus so that all these assumptions are met, allowing application software to rely on all communications taking place on time. This means that integration testing at the automotive manufacturer can concentrate on functional testing of the application software.

Another important benefit is that a large amount of communications protocol overhead can be avoided. Examples of how protocol overheads are reduced by obtaining timing guarantees are

- There is no need to provide frame acknowledgment within the communications layer, dramatically reducing bus traffic. The only case where an ECU can fail to receive a frame via CAN is if the ECU is "off the bus," a serious fault that is detected and handled by network management and on-board diagnostics.
- Retransmissions are unnecessary. The system level timing analysis guarantees that a frame will arrive on time. Timeouts only happen after a fault, which can be detected and handled by network management and/or the on-board diagnostics.

A Volcano system never suffers from intermittent overruns during correct operation because of the timing guarantees, and therefore achieves these efficiency gains.

19.3 Volcano Signals and the Publish/Subscribe Model

The Volcano system provides "signals" as the basic communication object. Signals are small data items that are sent between ECUs.

The "publish/subscribe" model is used for defining signaling needs. For a given ECU, there are a set of signals that are "published" (i.e., made available to the system integrator), and a number of "subscribed" signals (i.e., signals that are required as inputs to the ECU).

The signal model is provided directly to the programmer of ECU application software, and the Volcano software running in each ECU is responsible for translation between signals and CAN frames.

An important design requirement for the Volcano software was that the application programmer is unaware of the bus behavior: All the details of the network are hidden and the programmer only deals with signals through a simple API. This is crucial because a major problem with alternative techniques is that the application software makes assumptions about the CAN behavior and therefore changing the bus behavior becomes difficult.

In Volcano there are three types of signals:

- Integer signals. These represent unsigned numbers and are of a static size between 1 and 16 bits. So, for example, a 16 bit signal can store integers in the range 0–65,535.
- Boolean signals. These represent truth conditions (true/false). Note that this is not the same as a 1 bit integer signal (which stores the integer values 0 or 1).
- Byte signals. These represent data with no Volcano-defined structure. A byte signal consists of a fixed number between 1 and 8 bytes.

The advantage of Boolean and integer signals is that the values of a signal are independent of processor architecture (i.e., the values of the signals are consistent regardless of the "endianness" of the microprocessors in each ECU).

For published signals, Volcano internally stores the value of these signals and in case of periodic signals will send them to the network according to a pattern defined off-line by the system integrator. The system integrator also defines the initial value of a signal. The value of a signal persists until updated by the application program via a "write" call or until Volcano is reinitialized.

For subscribed signals, Volcano internally stores the current value of each signal. The system integrator also defines the initial value of a signal.

The value of a subscribed signal persists until

- It is updated by receiving a new value from the network, or
- Volcano is reinitialized, or
- A signal refresh timeout occurs and the value is replaced by a substitute value defined by the application programmer

In the case where new signal values are received from the network, these values will not be reflected in the values of subscribed signals until a Volcano "input" call is made.

A published signal value is updated via a write call. The latest value of a subscribed signal is obtained via a "read" call. A write call for a subscribed signal is not permitted.

The last-written value of a published signal may be obtained via a read call.

19.4 Update Bits

The Volcano concept permits placement of several signals with different update rates into the same frame. It provides a special mechanism named "update bit" to indicate which signals within the frame

has actually been updated, i.e., the ECU generating the signal wrote a fresh value of the signal since the last time it has been transmitted. The Volcano software on an ECU transmitting a signal automatically clears the update bit when it has been sent. This ensures that a Volcano-based ECU on the receiving side will know each time the signal has been updated (the application can see this update bit, by using flags tied to an update bit, see below). Using update bits to their full extent require that the underlying protocol is "secure." (Frames cannot be lost without being detected.) The CAN protocol is regarded as such, but not the local interconnect network (LIN) protocol. Therefore the update bit mechanism is limited to CAN within Volcano.

19.5 Flags

A flag is a Volcano object purely local to an ECU. It is bound to one of two things:

- The update bit of a received Volcano signal—the flag is set when the update bit is set
- The containing frame of a signal—the flag is set when the frame containing the signal is received (regardless of whether an update bit for the signal is set)

Many flags can be bound to each update bit, or the reception of a containing frame. Volcano sets all the flags bound to an object when the occurrence is seen. The flags are cleared explicitly by the application software.

19.6 Timeouts

A timeout is, like the flags, a Volcano object purely local to an ECU. The timeout is declared by the application programmer and is bound to a subscribed signal. A timeout condition occurs when the particular signal was not received within the given time limit. In this case, the signal (or/and a number of other signals) is/are set to a value specified as part of the declaration of the timeout. As with the flags, the timeout reset mechanism can be bound to either

- The update bit of a received Volcano signal
- The frame carrying a specific signal

19.7 Frames

A frame is a container capable of carrying a certain amount of data (0–8 bytes for CAN and LIN). Several signals can be packed into the available data space and transmitted together in one frame on the network. The total size of a frame is determined by the protocol. A frame can be transmitted periodically or sporadically. Each frame is assigned a unique "identifier." The identifier serves two purposes in the CAN case:

- Identifying and filtering a frame on reception at an ECU
- Assigning a priority to a frame

19.7.1 Immediate Frames

Volcano normally hides the existence of network frames from the application designer. However, under certain cases there is a need to send and receive frames with very short processing latencies. In these cases direct application support is required. Such frames are designated immediate frames.

There are two Volcano calls to handle immediate frames:

- A "transmit" call, which immediately sends the designated frame to the network
- A "receive" call, which immediately processes the designated incoming frame if that frame is pending

There is also a "read update bit" call to test the update bit of a subscribed signal within an immediate frame.

The signals packed into an immediate frame can be accessed with normal read and write function calls in the same way as all other normal signals.

The application programmer is responsible for ensuring that the transmit call is made only when the signal values of published signals are consistent.

19.7.2 Frame Modes

In Volcano it is allowed to specify different frame modes for an ECU. A frame mode is a description of an ECU working mode, where a set of frames (signals) can be active (input and output). The frames can be active in one or many frame modes. The timing properties of frames do not have to be the same for different frame modes supporting the same frame.

19.8 Network Interfaces

A network interface is the device used to send and receive frames to and from networks. A network interface connects a given ECU to the network. In the CAN case more than one network interface (CAN controller) on the same ECU may be connected to the same network. Likewise, an ECU may be connected to more than one network.

The network interface in Volcano is protocol specific. The protocols currently supported are CAN and LIN; FlexRay and MOST are under implementation.

The network interface is managed by a standard set of Volcano calls. These allow the interface to be initialized or reinitialized, connected to the network (i.e., begin operating the defined protocol), and disconnected from the network (i.e., take no further part in the defined protocol). There is also a Volcano call to return the status of the interface.

19.9 Volcano API

The Volcano API provides a set of simple calls to manipulate signals and to control the CAN/LIN controllers. There are also calls to control Volcano sending to, and receiving from networks. To manipulate signals there are read and write calls. A read call returns to the caller the latest value of a signal; a write call sets the value of a signal. The read and write calls are the same regardless of the underlying network type.

19.9.1 Volcano Thread-of-Control

There are two Volcano calls that must be called at the same fixed rate: v_input() and v_output(). If the v_gateway() function is used the same calling rate shall be used as for the v_input() and v_output() functions. The v_output() call places the frames into the appropriate controllers. The v_input() call takes received frames and makes the signal values available to read calls. The v_gateway() call copies values of signals in frames received from the network to values of signals in frames sent to the network. The v_sb_tick() call handles transmitting and receiving frames for sub-buses.

Volcano also provides a very low latency communication mechanism in the form of the *immediate frame API*. This is a "view" of frames on the network, which allows transmission and reception from/to the Volcano domain without the normal Volcano input/output latencies, or mutual exclusion requirements with the v_input() and v_output() calls. There are two communication calls in the immediate signal API: v_imf_rx() and v_imf_tx().

The v_imf_tx() call copies values of immediate signals into a frame and places the frame in the appropriate CAN controller for transmission. The v_imf_rx() takes a received frame containing immediate signals and makes the signal values available to read calls.

A third call v_imf_queued() allows the user to see if an immediate frame has really been sent on the network.

The controller calls allow the application to initialize, connect, and disconnect from networks, and to place the controllers into "sleep" mode among others.

19.10 Volcano Resource Information

The ambition of the Volcano concept is to provide a fully predictable communications solution. In order to achieve this, the resource usage of the Volcano embedded part has to be determined. Resources of special interest are memory and execution time.

19.11 Execution Time of Volcano Processing Calls

In order to bound processing time, a "budget" for the v_input() call, i.e., the maximum number of frames that will be processed by a single call to v_input(), has to be established. A corresponding process for transmitted frames applies as well.

19.12 Timing Model

The Volcano timing model covers end-to-end timing (i.e., from button press to activation). To be able to set in context, the signal timing information needed in order to analyze a network configuration of signals and frames, a timing model is used. This section defines the required information that must be provided by an application programmer in order to be able to guarantee the end-to-end timing requirements.

A Volcano signal is transported over a network within a frame. Figure 19.2 identifies six time points between the generation and consumption of a signal value.

The six time points are

1. Notional generation (signal generated), either by hardware (e.g., switch pressed) or software (e.g., timeout signaled). The user can define this point to best reflect their system.
2. First v_output() (or v_imf_tx() for an immediate frame) at which a new value is available. This is the first such call after the signal value is written by a write call.
3. The frame containing the signal is first entered for transmission (arbitration on a CAN bus).
4. Transmission of the frame completes successfully (i.e., the subscriber's communication controller receives the frame from the network).
5. v_input() (or v_imf_rx() for an immediate frame) makes the signal available to the application.
6. Notional consumption, the user application consumes the data. The user can define this point to best reflect their system.

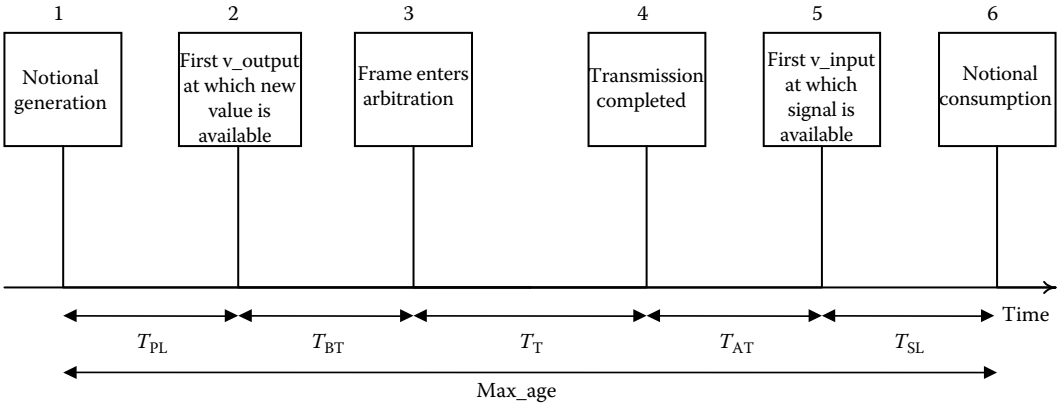


FIGURE 19.2 Volcano timing model.

The max_age of the signal is the maximum age, measured from notional generation, at which it is acceptable for notional consumption. The max_age is the overall timing requirement on a signal.

T_{PL} (publish latency) is the time from notional generation to the first v_output() call when the signal value is available to volcano (a write call has been made). It will depend on the properties of the publishing application. Typical values might be the frame_processing_period (if the signal is written fresh every period but this is not synchronized with v_output()), the offset between the write call and v_output() (if the two are synchronized), or the sum of the frame_processing_period and the period of some lower-rate activity which generates the value. This value must be given by the application programmer.

T_{SL} (subscribe latency), the time from the first v_input that makes the new value available to the application to the time when the value is "consumed." The consumption of a signal is a user-defined event which will depend on the properties of the subscribing function. As an example it can be a lamp being lit, or an actuator starting to move. This value must be given by the application programmer.

The intervals T_{BT}, T_T, and T_{AT} are controlled by the Volcano 5 configuration and are dependent upon the nature of the frame in which the signal is transported.

The value T_{BT} is the time before transmission (the time from the v_output call until the frame enters arbitration on the bus). T_{BT} is a per-frame value which depends on the type of frame carrying the signal (see later sections). This time is shared by all signals in the frame, and is common to all subscribers to those signals.

The value T_{AT} is the time after transmission (the time from when the frame has been successfully transmitted on the network until the next v_input call). T_{AT} is a per-frame value which may be different for each subscribing ECU.

The value T_T is the time required to transmit the frame (including the arbitration time) on the network.

19.13 Jitter

The application programmer at the supplier must also provide information of the "jitter" to the systems integrator. This information is as follows:

The input_jitter and output_jitter refer to the variability in the time taken to complete the v_input() and v_output() calls, measured relative to the occurrence of the periodic event causing Volcano processing to be done (i.e., calls to v_input(), v_gateway(), and v_output() to be made). Figure 19.3 shows how the output_jitter is measured:

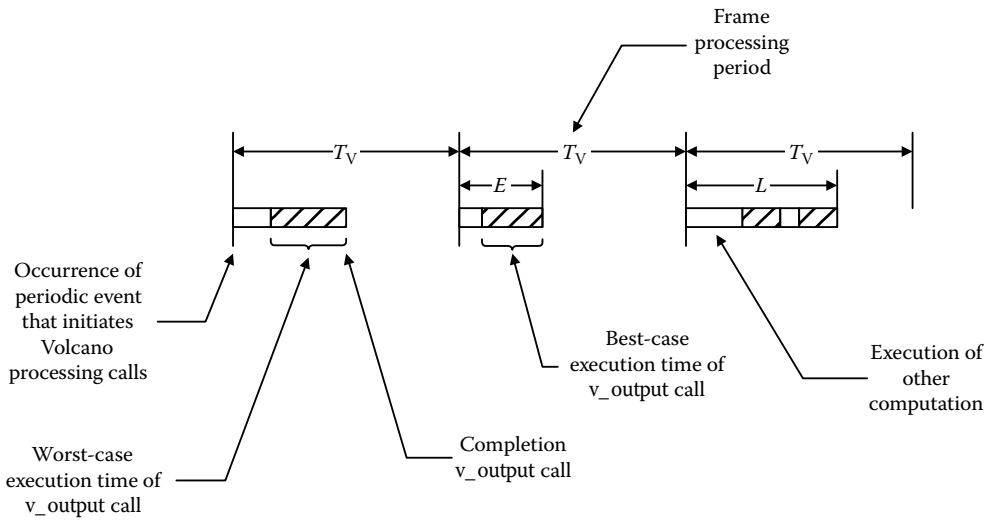


FIGURE 19.3 Output jitter.

In the figure, E marks the earliest completion time of the v_output() call, and L marks the latest completion time, relative to the start of the cycle. The output_jitter is therefore $L - E$. The input_jitter is measured according to the same principles.

If a single-thread system is used, without interrupts, the calculation of the input_jitter and output_jitter is straightforward: the earliest time is the best-case execution time of all the calls in the cycle (including the v_output() call), and the latest time is the worst-case execution time of all the calls. The situation is more complex if interrupts can occur or the system consists of multiple tasks, since the latest time must take into account preemption from interrupts and other tasks.

19.14 Capture of Timing Constraints

The declaration of a signal in a Volcano fixed configuration file provides syntax to capture the following timing-related information:

Whether a signals is **state** or **state change—(info_type)**

Whether a signals is **sporadic** or **periodic—(generation_type)**

The **latency**.

The **min_interval**.

The **max_interval**.

The **max_age**.

The first two (together with whether the signal is published or subscribed to) provide signal properties which determine the "kind" of the signal.

A **state** signal carries a value which completely describes the signaled property (e.g., the current position of a switch). A subscriber to such a signal need only observe the signal value when the information is required for the subscribers purposes (e.g., signal values can be "missed" without affecting the usefulness of later values).

A **state change** signal carries a value which must always be observed in order to be meaningful (e.g., distance traveled since last signal value). A subscriber must observe every signal value.

A **sporadic** signal is one which is written by the application in response to some event (e.g., a button press).

A **periodic** signal is one which is written by the application at regular intervals.

The **latency** of a signal is the time from notional generation to being available to Volcano (for a published signal), or from being made available to the application by Volcano to notional consumption (for a subscribed signal). Note that immediate signals (those in immediate frames) include time taken to move frames to/from the network in these latencies.

The **min_interval** has different interpretation for published and for subscribed signals.

For a published signal, it is the minimum time between any pair of write calls to the signal (this allows, e.g., the calculation of the maximum rate at which the signal could cause a sporadic frame carrying it to be transmitted).

For a subscribed signal, it is the minimum acceptable time between arrivals of the signal. This is optional—it is intended to be used if the processing associated with the signal is triggered by arrival of a new value, rather than periodic. In such a case, it provides a constraint that the signal should not be connected to published signal with a faster rate.

The **max_interval** has different interpretation for published and for subscribed signals.

For a published signal, the interesting timing information is already captured by min_interval and publish latency.

For a subscribed signal it is the maximum interval between "notional consumptions" of the signal (i.e., it can be used to determine that signal values are sampled quickly enough that none will be missed).

The **max_age** of a signal is the maximum acceptable age of a signal at notional consumption, measured from notional generation. This value is meaningful for subscribed signals.

In addition to the signal timing properties described above, the Volcano fixed configuration file provides syntax to capture the following additional timing-related information:

- Volcano processing period
- Volcano jitter time

The *volcano processing period* defines the nominal interval between successive v_input() calls on the ECU, and also between successive v_output() calls (i.e., the rates of the calls are the same, but v_input() and v_output() are not assumed to "become due" at the same instant). For example, if the volcano processing period is 5 ms then each v_output() call becomes due 5 ms after the previous one becomes due.

The *volcano jitter* defines the time by which the actual call may lag behind the time at which it became due. Note that becomes due refers to the start of the call, and jitter refers to completion of the call.

19.15 Volcano Network Architect

To manage increasing complexity in electrical architectures, a structured development approach is believed essential to assure correctness by design. Volcano automotive group has developed a network design tool, volcano network architect (VNA), to support a development process, based on strict systems engineering principles. Gatewaying of signals between different networks is automatically handled by the VNA tool and the accompanying embedded software.

The tool supports partitioning of responsibilities into different roles such as system integrator and function owner. Third party tools may be used for functional modeling. These models can be imported into VNA.

VNA is the top-level tool in the Volcano automotive groups tool chain for designing vehicle network systems. The tool chain supports important aspects of systems engineering such as

- Use of functional modeling tools
- Partitioning of responsibilities
- Abstracting away from hardware and protocol-specific details providing a signal-based API for the application developer
- Abstracting away from the network topology through automatic gatewaying between different networks
- Automatic frame compilation to ensure that all declared requirements are fulfilled (if possible), that is, delivering correctness by design
- Reconfiguration flexibility by supporting post-compile-time reconfiguration capability

The VNA tool supports network design and makes management and maintenance of distributed network solutions more efficient. The tool support capturing of requirements and then takes the user through all stages of network definition.

19.15.1 Car OEM Tool Chain: One Example

Increasing competition and complex electrical architectures demands enhanced processes. Function modeling has proved to be a suitable tool to capture the functional needs in a vehicle. Tools such as Rational Rose provide a good foundation to capture all different functions and other tools (Statemate, Simulink) model them in order to allocate objects and functionality in the vehicle. Networking is essential since the functionality is distributed among a number of ECUs in the vehicle. Substantial parts of the outcome from the function modeling are highly suitable to use as input to a network design tool such as VNA.

The amount of information required to properly define the networks are vast. To support input of data, VNA provides an automated import from third party tools through an XML-based format.

It is the job of the signal database administrator/system integrator to ensure that all data entered into the system are valid and internally consistent. VNA supports this task through a built-in multilevel consistency checker that verifies all data.

In this particular approach the network is designed by the system integrator in close contact with the different function owners in order to capture all necessary signaling requirements—functional and nonfunctional (including timing). When the requirements are agreed and documented in VNA, the system integrator uses VNA to pack all signals into frames, this can be done manually or automatically. The algorithm used by VNA handles gatewaying by partitioning end-to-end timing requirements into requirements per network segment.

All requirements are captured in the form of a Microsoft Word document called software requirement specification (SWRS) that is generated by VNA and sent to the different node owners as a draft copy to be signed off. When all SWRS has been signed off, VNA automatically creates all necessary configuration files used in the vehicle along with a variety of files for third party analysis and measurement tools.

The network level (global) configuration files are used as input to the Volcano configuration tool and Volcano back-end tool in order to generate a set of downloadable binary configuration files for each node. The use of reconfigurable nodes makes the system very flexible since the Volcano concept separates application-dependent information and network-dependent information. A change in the network by the system integrator can easily be applied to a vehicle without having to recompile the application software in the nodes. The connection between function modeling and VNA provide good support for iterative design. It verifies network consistency and timing up front, to ensure a predictable and deterministic network.

19.16 VNA: Tool Overview

19.16.1 Global Objects

The workflow in VNA ensures that all relevant information about the network is captured. Global objects shall be created first, and then (re)used in several projects. The VNA user works with objects of types such as signals, nodes, interfaces, etc. These objects are used to build up the networks used in a car. Signals are defined by name and type, and can have logical or physical encoding information attached. Interfaces detailing hardware requirements are defined leading to describe actual nodes on a network. For each node, receive and transmit signals are defined, and timing requirements are provided for the signals. This information is intended for "global use," that is, across car variants, platforms, etc.

19.16.2 Project or Configuration Related Data (Projects, Configurations, Releases)

When all global data have been collected, the network will be designed by connecting the interfaces in a desired configuration. VNA has strong project and variant handling. Different configurations can selectively use or adapt the global objects, for example, by removing a high-end feature from a low-end car model. This means that VNA can manage multiple configurations, designs, and releases, with version and variant handling.

The release handling ensures that all components in a configuration are locked. It is, however, still possible to reuse the components in unchanged form. This makes it possible to go back to any released configuration at any point in time.

The database is a central part of the VNA system. In order to ensure highest possible performance, each instance of VNA accesses a local mirror of the database which is continuously synchronized with its parent (Figure 19.4).

19.16.3 Database

All data objects, both global and configuration specific, are stored in a common database. The VNA tool was designed to have one common multiuser database per car OEM. In order to secure highest possible performance, all complex and time-consuming VNA operations are performed toward a local RAM mirror of the database. A specially designed database interface ensures consistency in the local mirror. Operations that are not time critical, such as database management, operate toward the database.

The built-in multiuser functionality allows multiple users to access all data stored in the database simultaneously. To ensure that a data object is not modified by more than one user, the object must be locked before any modification, read access is of course allowed for all users although an object is locked for modification.

19.16.4 Version and Variant Handling

The VNA database implements functionality for variants and versions handling. Most of the global data objects, e.g., signals, functions, and nodes, may exist in different versions, but only one version of an object can be used in a specific project/configuration.

The node objects can be seen as the main global objects, since hierarchically they include all other types of global objects. The node objects can exist in different variants but only one object can be used from a variant folder in a particular project/configuration.

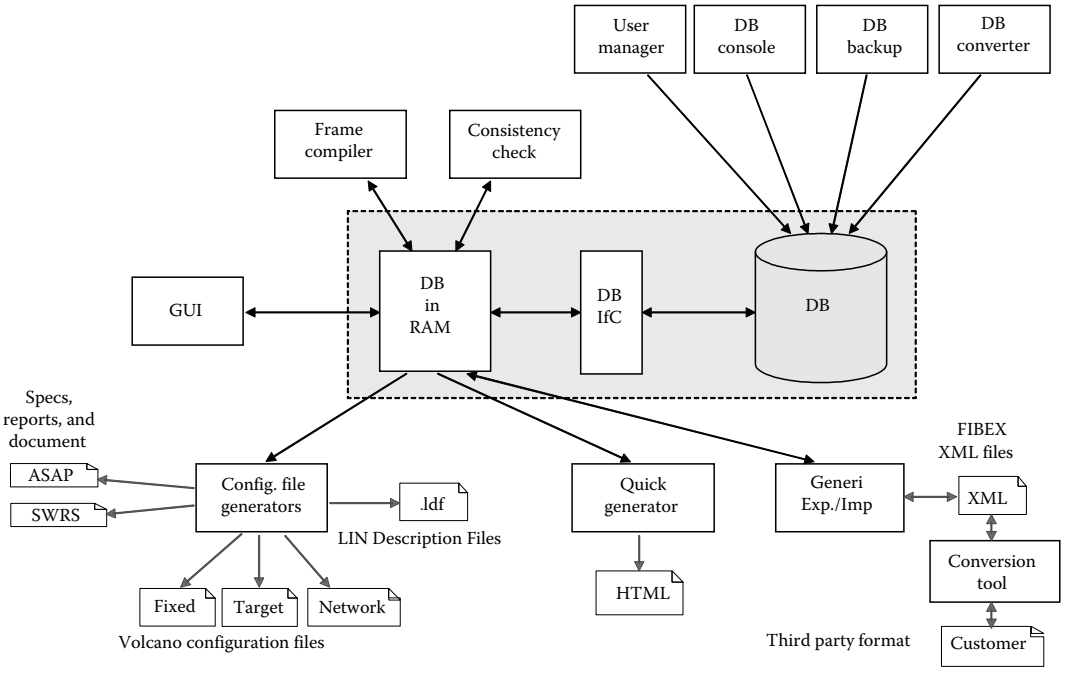


FIGURE 19.4 Structure of the VNA tool.

19.16.5 Consistency Checking

Extensive functionality for consistency checking is built into the VNA tool. The consistency check can be manually activated when needed, but is also running continuously to check user input and give immediate feedback on any suspected inconsistency. The consistency check ensures that the network design follows predefined rules and generates errors when appropriate.

19.16.6 Timing Analysis/Frame Compilation

The Volcano concept is based on a foundation of guaranteed message latency and a signal-based publish and subscribe model. This provides abstraction by hiding the network and protocol details, allowing the developer to work in the application domain with signals, functions, and related timing information.

Much effort has been spent on developing and refining the timing analysis in VNA. The timing analysis is built upon a scheduling model called DMA, deadline monotonic analysis, and calculates the worst-case latency for each frame among a defined set of frames sent on the bus. Parts of this functionality have been built into the consistency check routine as described above but the real power of the VNA tool is found in the frame packer/frame compiler functionality.

The frame packer/compiler attempts to create an optimal packing of signals into frames, than calculate the proper IDs to every frame ensuring that all the timing requirements captured earlier in the process are fulfilled (if possible). This automatic packing of multiple signals into each frame makes more efficient use of the data bus, by amortizing some of the protocol overheads involved thus lowering bus load. The combined effect of multiple signals per frame and perfect filtering results in a lower interrupt and CPU load which means that the same performance can be obtained at lower cost. The frame packer can create the most optimal solution if all nodes are reconfigurable. To handle carry over nodes that are not reconfigurable (ROM-based), these nodes and their associated frames can be

classed as "fixed." Frame packing can also be performed manually if desired. Should changes to the design be required at a later time, the process allows rapid turnaround of design changes, rerunning the frame compiler, and regenerating the configuration files.

The VNA tool can be used to design network solutions that are later realized by embedded software from any provider. However, the VNA tool is designed with the volcano embedded software (volcano target package [VTP]) in mind, which implements the expected behavior into the different nodes. To get the full benefits of the tool chain, VNA and VTP should be used together.

19.16.7 Volcano Filtering Algorithm

A crucial aspect of network configuration is how to choose identifiers so that the load on a CPU related to handling of interrupts generated by frames of no interest for the particular node is minimized—most CAN controllers have only limited filtering capabilities. The Volcano filtering algorithm is designed to achieve this.

An identifier is split into two parts: the priority bits and the filter bits. All frames on a network must have unique priority bits; for real-time performance the priority setting of a frame should reflect the relative urgency of the frame. The filter bits are used to determine if a CAN controller should accept or reject a frame. Each ECU that needs to receive frames by interrupts is assigned a single filter bit; the hardware filtering in the CAN controller is set to "must match 1" for the filter bit, and "don't care" for all other bits.

The filter bits of a frame are set for each ECU to which the frame needs to be seen. So a frame that is broadcast to all ECUs on the network is assigned filter bits all set to "1." For a frame sent to a single ECU on the network just one filter bit is set. Figure 19.5 below illustrates this; the frame shown is sent to four ECUs.

Figure 19.5 shows an example of a CAN identifier on an extended CAN network. The network clause has defined the CAN identifiers to have 7 priority bits, and 13 filter bits. The least-significant bit of the value corresponds with the bit of the identifier transmitted last. Only legal CAN identifiers can be specified: identifiers with the seven most-significant bits equal to "1" are illegal according to the CAN standard.

If an ECU takes an interrupt for just the frames that it needs, then the filtering is said to be *perfect*. In some systems there may be more ECUs needing to receive frames by interrupt than there are filter bits in the network; in this case, some ECUs will need to share a bit. If this happens then Volcano will filter the frames in software, using the priority bits to uniquely identify the frame and discarding unwanted frames.

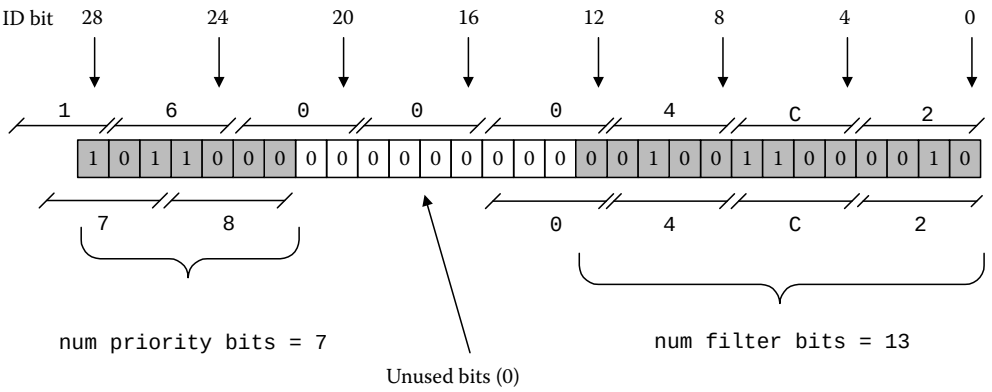


FIGURE 19.5 CAN ID filtering scheme in volcano.

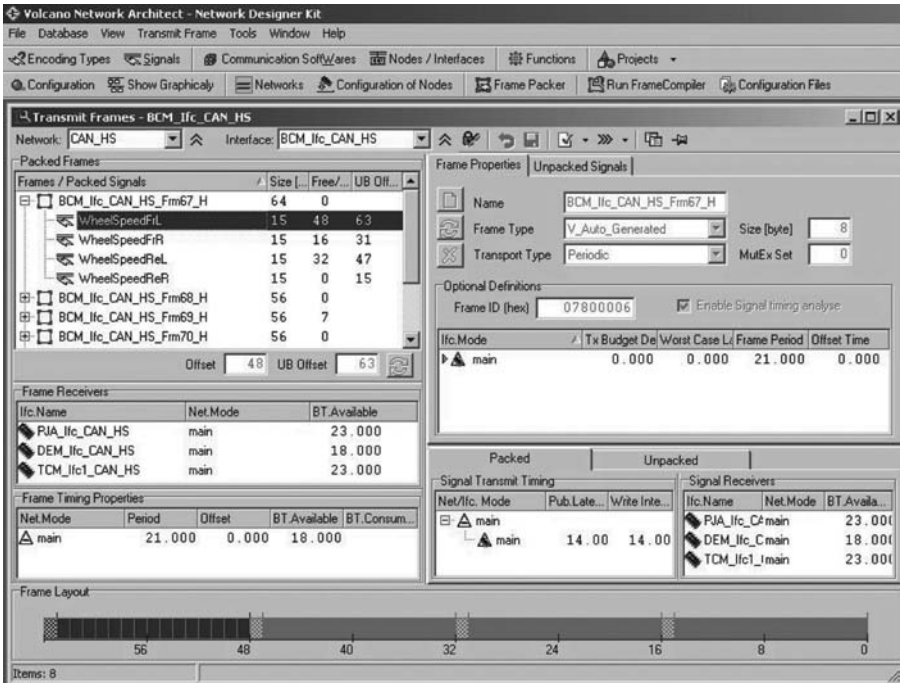

Volcano Network Architect - Network Designer Kit
File Database View Transmit Frame Tools Window Help
Encoding Types Signals Communication Softwares Nodes / Interfaces Functions Projects
Configuration Show Graphicaly Networks Configuration of Nodes Frame Packer Run FrameCompiler Configuration Files
Transmit Frames - BCM_Ifc_CAN_HS
Network: CAN_HS Interface: BCM_Ifc_CAN_HS
Packed Frames
Frames / Packed Signals Size [... Free/... UB Off...
BCM_Ifc_CAN_HS_Frm67_H 64 0
WheelSpeedFrL 15 48 63
WheelSpeedFrR 15 16 31
WheelSpeedReL 15 32 47
WheelSpeedReR 15 0 15
BCM_Ifc_CAN_HS_Frm68_H 56 0
BCM_Ifc_CAN_HS_Frm69_H 56 7
BCM_Ifc_CAN_HS_Frm70_H 56 0
Offset 48 UB Offset 63
Frame Receivers
Ifc.Name Net.Mode BT.Available
PJA_Ifc_CAN_HS main 23.000
DEM_Ifc_CAN_HS main 18.000
TCM_Ifc1_CAN_HS main 23.000
Frame Timing Properties
Net.Mode Period Offset BT.Available BT.Consum...
main 21.000 0.000 18.000
Frame Properties Unpacked Signals
Name BCM_Ifc_CAN_HS_Frm67_H
Frame Type V_Auto_Generated Size [byte] 8
Transport Type Periodic MultEx Set 0
Optional Definitions
Frame ID (hex) 07800006 Enable Signal timing analyse
Ifc.Mode Tx Budget De Worst Case L(Frame Period Offset Time
main 0.000 0.000 21.000 0.000
Packed Unpacked
Signal Transmit Timing
Net/Ifc. Mode Pub.Late... Write Inte...
main
main 14.00 14.00
Signal Receivers
Ifc.Name Net.Mode BT.Availa...
PJA_Ifc_CA main 23.00(
DEM_Ifc_C main 18.00(
TCM_Ifc1_I main 23.00(
Frame Layout
56 48 40 32 24 16 8 0
Items: 8

FIGURE 19.6 VNA screen.

The priority bits are the most significant bits. They indicate priority and uniquely identify a frame. The number of priority bits must be large enough to uniquely identify a frame in a given network configuration. The priority bits for a given frame are set by the relative urgency (or deadline) of the frame. This is derived from how urgently each subscriber of a signal in the frame needs the signal (as described earlier). In most systems 5–10 priority bits are sufficient.

The filter bits are the remaining least-significant bits and are used to indicate the destination ECUs for a given frame. Treating them as a "target mask" does this: Each ECU (or group of ECUs) is assigned a single filter bit. The filtering for a CAN controller in the ECU is set up to accept only frames where the corresponding filter bit in the identifier is set. This can give "perfect filtering": an interrupt is raised if and only if the frame is needed by the ECU. Perfect filtering can dramatically reduce the CPU load compared to filtering in software. Indeed, perfect filtering is essential if the system integrator needs to connect ECUs with slow 8 bit CPUs to high-speed CAN networks (if filtering were implemented in software the CPU would spend most of its available processing time handling interrupts and discarding unwanted frames). The filtering scheme also allows for broadcast of a frame to an arbitrary set of ECUs. This can reduce the traffic on the bus since frames do not need to be transmitted several times to different destinations (Figure 19.6).

Because the system integrator is able to define the configuration data and because that data defines the complete network behavior of an ECU, the in-vehicle networks are under the control of the system integrator.

19.16.8 Multiprotocol Support

The existing version of VNA supports the complementary, contemporary network protocols of CAN and LIN. As network technology continues to advance into other protocols, VNA will also move to support these advances. The philosophy behind is that communications has to be managed in

one single development environment, covering all protocols used, in order to ensure end-to-end timing predictability, still providing the necessary architectural freedom to choose the most economic solution for the task.

19.16.9 Gatewaying

A network normally consists of multiple network segments using different protocols. Signals may be transferred from one segment to another through a gateway node. As implemented throughout the whole tool chain of Volcano automotive group, gatewaying of data even across multiple protocols is automatically configured in VNA. In this way, VNA allow any node to subscribe any signal generated on any network without needing to know how this signal is gatewayed from the publishing node. Handling of timing requirements over one or more gateways is also handled by VNA. The Volcano solution requires no special gatewaying hardware and therefore provides the most cost-efficient solution to signal gatewaying.

19.16.10 Data Export and Import

The VNA tool enables the OEMs to get a close integration between VNA and functional modeling tools and to share data between different OEMs and subcontractors, e.g., node developers.

Support of emerging standards such as FIBEX and XML will further simplify information sharing and become a basis for configuration of third party communication layers.

Volcano tool chain includes networking software running in each ECU in the system. This software uses the configuration data to control the transmission and reception of frames on one or more buses and present signals to the application programmer. One view of the Volcano network software is as a "communications engine" under the control of the system integrator. The view of the application programmer is different: the software is a black box into which published signals are placed, and out of which can be summoned subscribed signals.

The main implementation goals for Volcano target software are as follows:

- Predictable real-time behavior, no data-loss under any circumstances
- Efficiency (low RAM usage, fast execution time, small code size)
- Portability (low cost of moving to a new platform)

19.17 Volcano Configuration

Building a configuration is a key part of the Volcano concept. A configuration is, as already mentioned, based around details, such as how signals are mapped into frames allocation of identifiers, and processing intervals.

For each ECU, there are two authorities acting in the configuration process: the system integrator and the ECU supplier. The system integrator provides the Volcano configuration for the ECU regarding the network behavior at the system level, and the supplier provides the Volcano configuration data for the ECU in terms of the internal behavior.

19.18 Configuration Files

The Volcano configuration data is captured in four different types of files. These are

- *Fixed information* (agreed between the supplier and system integrator).
- *Private information* provided by the ECU supplier. The ECU supplier does not necessarily have to provide this information to the system integrator.

- *Network configuration information* supplied by the system integrator.
- *Target information* (supplier description of the ECU published to the system integrator).

19.18.1 Fixed Information

The fixed information is the most important in achieving a working system. It consists of a complete description of the dependencies between the ECU and the network. This includes a description of the signals the ECU needs from the network, how often Volcano calls will be executed, and so on. The information also includes description of the CAN controller(s), and possible limitations regarding reception and transmission boundaries and supported frame modes. The fixed information forms a "contract" between the supplier and the system integrator: the information should not be changed without both parties being aware of the changes. The fixed information file is referred to as the "FIX" file.

19.18.2 Private Information

The private file contains additional information for Volcano, which does not affect the network: timeout values associated to signals and what flags are used by the application. The private information file is referred to as the "PRI" file.

19.18.3 Network Information

The network information specifies the network configuration of the ECU. The system integrator must define the number of frames sent from and received by the ECU, the frame identifier and length, and details of how the signals in the agreed information are mapped into these frames. Here, the vehicle manufacturer also defines the different frame modes used in the network. The network information file is referred to as the "NET" file.

19.18.4 Target Information

The target information contains information about the resources that the supplier has allocated to Volcano in the ECU. It describes the ECU's hardware (e.g., used CAN controllers and where those are mapped in memory). The target information file is referred to as the "TGT" file.

19.19 Workflow

The Volcano system identifies two major roles in the development of a network of ECUs: the application designer (which may include the designer of the ECU system or the application programmer) and the system integrator. The application designer is typically located at the organization developing the ECU hardware and application software. The system integrator is typically located at the vehicle manufacturer. The interface between the application designer and the system integrator is carefully controlled, and the information owned by each side is strictly defined. The Volcano tool chain implementation is clearly reflecting this partitioning of roles.

The Volcano system includes a number of tools to help the system integrator in defining a network configuration. The "Network Architect" is a high level design tool, with a database containing all the publish/subscribe information for each ECU available, as described in the previous sections. After mapping the signaling needs on particular network architecture, thus defining the connections between the published and subscribed signals, an automatic frame compiler will be run. The "frame compiler" tool uses the requirements captured earlier to build a configuration which meet

those requirements. There are many possibilities to optimize the bus behavior. The frame compiler includes the CAN bus timing analysis and LIN schedule table generation and will not generate a configuration that violates the timing requirements placed on the system. The frame compiler also uses the analysis to answer "what if?" type of questions and guide the user in building a valid and optimized network configuration.

The output of the frame compiler is used to build configuration data specific to each ECU. This is used by the Volcano target software in the ECU to properly configure and use the hardware resources.

The Volcano configuration data generator tool set (V5CFG/V5BND) is used to translate this ASCII text information to executable binary code in the following way:

- When the supplier executes the tool, it reads the FIX, PRI, and TGT files to generate compile-time data files. These data files are compiled and linked together with the application program together with the Volcano library supplied for the specific ECU system.

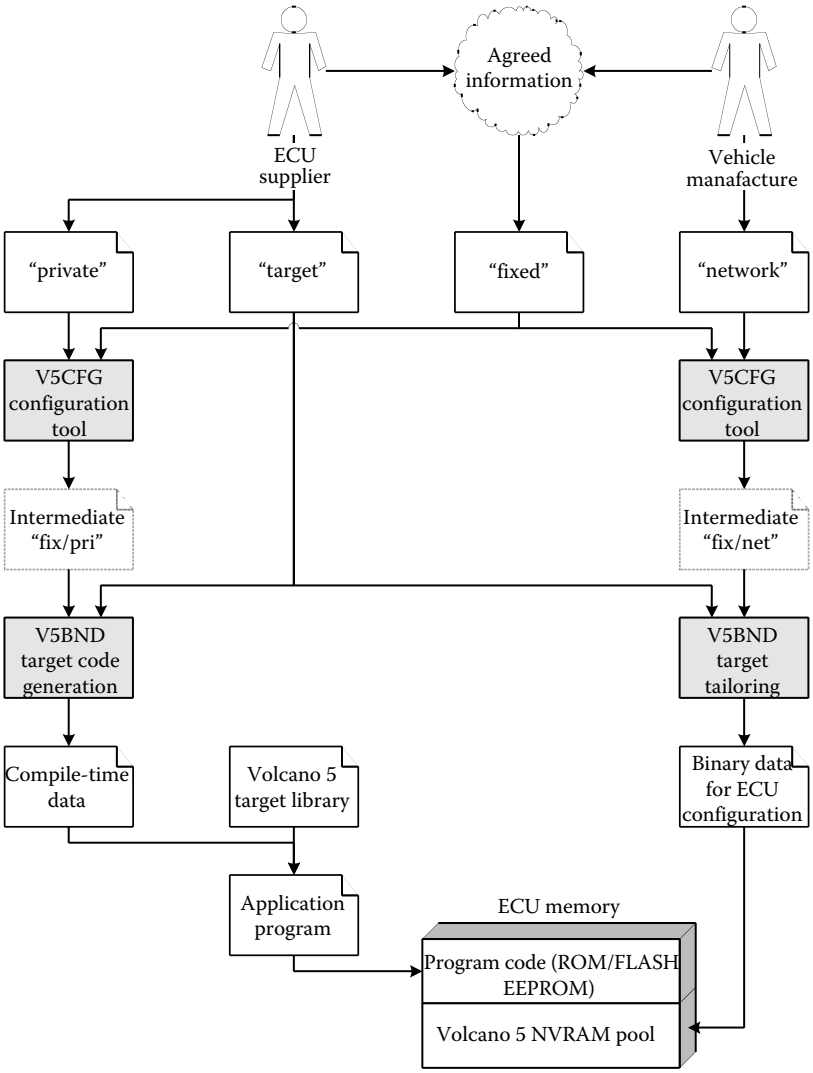


FIGURE 19.7 Configuration of Volcano target software.

- When the vehicle manufacturer executes the tool, it reads the FIX, NET, and TGT files to generate the binary data that is to be located in the ECU's Volcano configuration memory (known as the Volcano NVRAM). An ECU is then configured (or reconfigured) by downloading the binary data to the ECU's memory.

Note: It is vital to realize that changes to either the FIX or the TGT file cannot be done without having coordination between the system integrator and the ECU supplier.

The vehicle manufacturer can, however, change the NET file without informing the ECU supplier. In the same way, the ECU supplier can change the PRI file without informing the system integrator. Figure 19.7 shows how the Volcano Target Code for an ECU is configured by the supplier and the system integrator:

The Volcano concept and related products has been successfully used in production since 1996. Present car OEMs using the entire tool chain are Aston Martin, Jaguar, Land Rover, MG Rover, SAIC, Volvo Cars, and Volvo Bus Corporation. After a lengthy evaluation, Airbus decided to use VNA for design and verification of on-board CAN networks in aircraft in 2007.

Acknowledgment

I wish to acknowledge the contributions of István Horváth and Niklas Amberntsson at Mentor Graphics Corp. for their contributions to this chapter.

Reference

1. K. Tindell and A. Burns, Guaranteeing message latencies on controller area network (CAN), *Proceedings of the 1st International CAN Conference*, 1994, pp. 2–11.

Bibliography

1. K. Tindell, H. Hansson, and A. J. Wellings, Analysing real-time communications: Controller area network (CAN), *Proceedings of the 15th IEEE Real-Time Systems Symposium*, San Juan, Puerto Rico, 1994, pp. 259–265.
2. K. Tindell, A. Rajnak, and L. Casparsson, CAN communications concept with guaranteed message latencies, SAE Paper, 1998.
3. L. Casparsson, K. Tindell, A. Rajnak, and P. Malmberg, Volcano—A revolution in on-board communication, Volvo Technology Report, 1998.
4. W. Specks and A. Rajnák, The scaleable network architecture of the Volvo S80, *8th International Conference on Electronic Systems for Vehicles*, Baden–Baden, Germany, October 1998, pp. 597–641.

More Information

1. http://www.mentor.com/solutions/automotive

IV

Networked Embedded Systems in Industrial Automation

Field Area Networks in Industrial Automation

20 Fieldbus Systems: Embedded Networks for Automation *Thilo Sauter* **20**-1
Introduction • What Is a Fieldbus? • History • Communication Fundamentals—The OSI Model • Fieldbus Characteristics • Networking Networks—Interconnection in Heterogeneous Environments • Industrial Ethernet—The New Fieldbus • Aspects for Future Evolution • Appendix

21 Real-Time Ethernet for Automation Applications *Max Felser* **21**-1
Introduction • Structure of the IEC Standardization • Real-Time Requirements • Practical Realizations • Summary: Conclusions

22 Configuration and Management of Networked Embedded Devices *Wilfried Elmenreich* **22**-1
Introduction • Concepts and Terms • Requirements on Configuration and Management • Interface Separation • Profiles, Datasheets, and Descriptions • Application Development • Configuration Interfaces • Management Interfaces • Maintenance in Fieldbus Systems • Conclusion • Acknowledgment

Wireless Network Technologies in Industrial Automation

23 Networked Control Systems for Manufacturing: Parameterization, Differentiation, Evaluation, and Application *James R. Moyne and Dawn M. Tilbury* **23**-1
Introduction • Parameterization of Industrial Networks • Differentiation of Industrial Networks • NCS Characterization • Applications for Industrial Networks • Future Trends • Acknowledgments

24 Wireless LAN Technology for the Factory Floor: Challenges and Approaches *Andreas Willig* **24**-1
Introduction • Wireless Industrial Communications and Wireless Fieldbus: Challenges and Problems • Wireless LAN Technology and Wave Propagation • Physical Layer: Transmission Problems and Solution Approaches • Problems and Solution Approaches on the MAC and Link Layer • Wireless Fieldbus Systems: State of the Art • Wireless Ethernet/IEEE 802.11 • Summary

25 Wireless Local and Wireless Personal Area Network Communication in Industrial Environments *Kirsten Matheus* **25**-1
Introduction • WLAN, WPAN, WSN, Cellular and Ad Hoc Networks • Bluetooth Technology • IEEE 802.11 • IEEE 802.15.4/ZigBee • Coexistence of WPAN and WLAN • Summary and Conclusions

26 Hybrid Wired/Wireless Real-Time Industrial Networks *Gianluca Cena, Adriano Valenzano, and Stefano Vitturi*..... **26**-1
Introduction • Relevant Industrial Networks • Implementation of Hybrid Networks • IEEE 802.11-Based Extensions: Fieldbuses • IEEE 802.11-Based Extensions: RTE Networks • IEEE 802.15.4-Based Extensions • Conclusions

27 Wireless Sensor Networks for Automation *Jan-Erik Frey and Tomas Lennvall* **27**-1
Introduction • WSN in Industrial Automation • Development Challenges • Reference Case • Communication Standards • Low-Power Design • Packaging • Modularity • Power Supply

28 Design and Implementation of a Truly-Wireless Real-Time Sensor/Actuator Interface for Discrete Manufacturing Automation *Guntram Scheible, Dacfey Dzung, Jan Endresen, and Jan-Erik Frey*..... **28**-1
Introduction • WISA Requirements and System Specifications • Communication Subsystem Design • Communication Subsystem Implementation • Wireless Power Subsystem • Practical Application and Performance Comparison • Summary

20

Fieldbus Systems: Embedded Networks for Automation

20.1 Introduction **20**-1
20.2 What Is a Fieldbus? **20**-3
20.3 History **20**-6
Roots of Industrial Networks • Evolution of Fieldbusses
20.4 Communication Fundamentals—The OSI Model **20**-10
Layer Structure • Communication Services • Quality-of-Service Parameters
20.5 Fieldbus Characteristics **20**-20
Traffic Characteristics and Requirements • Fieldbus Systems and the OSI Model • Network Topologies • Medium Access Control • Communication Paradigms • Above the OSI Layers—Interoperability and Profiles • Fieldbus Management • Quest for Unification—The NOAH Approach
20.6 Networking Networks—Interconnection in Heterogeneous Environments **20**-45
Protocol Tunneling • Gateways
20.7 Industrial Ethernet—The New Fieldbus **20**-51
20.8 Aspects for Future Evolution **20**-55
20.9 Appendix **20**-56
References **20**-59

Thilo Sauter
Austrian Academy of Sciences

20.1 Introduction

Few developments have changed the face of automation so profoundly as the introduction of networks did. Especially, fieldbus systems—networks devised for the lowest levels of the automation hierarchy—had an enormous influence on the flexibility and performance of modern automation systems in all application areas. However, fieldbus systems were not the result of some "divine spark," but they emerged in a continuous and often cumbersome evolution process. Today, many applications areas are unthinkable without them: factory automation, distributed process control, building and home automation, substation automation and more generally energy distribution, but also in-vehicle networking, railway applications, and avionics. All these fields heavily rely on the availability of appropriate networks accounting for the special demands of the individual application.

But, what exactly is a fieldbus? Even after a quarter of a century of fieldbus development, there exists no clear-cut definition for the term. The "definition" given in the International Electrotechnical Commission (IEC) 61158 fieldbus standard is more a programmatic declaration or a least common multiple compromise than a concise formulation [1]: "A fieldbus is a digital, serial, multidrop, data bus for communication with industrial control and instrumentation devices such as—but not limited to—transducers, actuators, and local controllers." It comprises some important characteristics, but is far from being complete. On the other hand, it is a bit too restrictive.

A more elaborate explanation is given by the Fieldbus Foundation, the user organization supporting one of the major fieldbus systems [2]: "A Fieldbus is a digital, two-way, multidrop communication link among intelligent measurement and control devices. It serves as a local area network (LAN) for advanced process control, remote input/output (I/O), and high speed factory automation applications." Again, this is a bit restrictive, for it limits the application to process and factory automation, the primary areas where the Foundation Fieldbus is being used.

The lack of a clear definition is mostly due to the complex evolution history of fieldbusses. In most cases, bus systems emerged primarily to break up the conventional star-type point-to-point wiring schemes connecting simple digital and analog I/O devices to central controllers, thereby laying the grounds for the implementation of really distributed systems with more intelligent devices. As it was declared in the original mission statement of the IEC work, "the Field Bus will be a serial digital communication standard which can replace present signaling techniques such as 4–20 mA . . . so that more information can flow in both directions between intelligent field devices and the higher-level control systems over shared communication medium. . ." [3,4]. Still in more recent publications this aspect is seen as the only raison d'être for fieldbus systems [5], which is, however, short-sighted and does not do the fieldbus justice.

Today, fieldbus systems comprising communication networks and devices would likely be called embedded networks or networked embedded systems. In the 1980s, when their era began, these terms were still unknown. Yet, their main features and development stimuli are comparable to what is behind the modern concepts, which are by the way similarly weakly defined:

- *Focused solutions*. Like embedded systems and their networks, fieldbus systems were no general-purpose developments, even if they were said to be. They were always developed with a concrete application field in mind and designed to meet the respective boundary conditions (like temporal behavior, efficiency, reliability, but also cost) in the best possible way.
- *Smart devices*. An essential objective for embedded systems and fieldbusses alike is to bring more intelligence to the field, i.e., to the end devices. Like in embedded systems, fieldbus developers also used the technological building blocks available at the time where possible, such as standard microcontrollers to keep costs low. However, if special needs were to be met, also dedicated solutions were devised.
- *Limited resources*. Both embedded applications and fieldbus systems share the fundamental problem that resources are limited. No matter what the state of art in microelectronics is, embedded devices (and field devices) are less powerful than standard computers. Communication (sub)systems usually have less available bandwidth than computer networks, and power consumption is an issue.
- *Comprehensive concepts*. Fieldbus systems are not just networks. Communication is only part of a distributed automation concept with comprehensive application software and tool chains. In some advanced cases, fieldbusses were embedded into special frameworks exhibiting many characteristics of distributed operating systems. This is a typical feature also in modern networked embedded systems.

- *Distribution*. A network is the prerequisite of distributed systems, many data-processing tasks can be removed from a central controller and placed directly in the field devices if they are sufficiently smart and the interface can handle reasonably complex ways of communication.
- *Flexibility and modularity*. A fieldbus installation like any other network can be extended much more easily than a centralized system, provided the limitations of addressing space, cable length, etc. are not exceeded. For the special case of fieldbusses, simplification of the parameterization and configuration of complex field devices is an additional benefit making system setup and commissioning easier.
- *Maintainability*. Monitoring of devices, applying updates, and other maintenance tasks are easier, if at all possible, via a network.

All these aspects show that apart from the names, there is not so much difference in the general ideas of fieldbusses and embedded networks. Still there are peculiarities in the history as well as in the technical characteristics of fieldbus systems that are worth to be looked at and that might be interesting also for contemporary networked embedded systems. The purpose of this chapter thus is to give an overview of the nature of fieldbus systems. It briefly reviews the evolution from the historical roots up to the tedious standardization efforts. It discusses in detail typical characteristics that distinguish fieldbusses from other types of networks. Current activities aiming at using Ethernet in automation are reviewed, and evolution prospects are given. It appears that fieldbus systems have reached their climax, and that future years will bring a mix of Ethernet-/Internet-based solutions and still-to-be-developed, new field-level networks.

20.2 What Is a Fieldbus?

As said in the introduction, fieldbus systems have to be seen as an integrative part of a comprehensive automation concept and not as standalone solutions. The name is therefore programmatic and evocative. Interestingly enough, not even the etymology of the term itself is fully clear. The English word "fieldbus" is definitely not the original one. It appeared around 1985 when the fieldbus standardization project within IEC TC65 was launched [4] and seems to be a straightforward literal translation of the German term "Feldbus," which can be traced back until about 1980 [6]. Indeed, the overwhelming majority of early publications in the area is available only in German. The word itself was coined in process industry and primarily refers to the process field, designating the area in a plant where lots of distributed field devices, mostly sensors and actuators are in direct contact with the process to be controlled. Slightly after the German expression and sharing its etymological root, the French word "réseau de terrain" (or "réseau d'instrumentation," instrumentation network) emerged. This term was not specifically targeted at the process industry, but refers also to large areas with scattered devices. The connection of such devices to the central control room was traditionally made via point-to-point links, which resulted in a significant and expensive cabling need. The logical idea, powered by the advances of microelectronics in the late 1970s, was to replace this starlike cabling in the field by a party-line, bus-like installation connecting all devices via a shared medium—the fieldbus [7,8].

Given the large dimensions of process automation plants, the benefits of a bus are particularly evident. However, the concept was not undisputed when it was introduced. The fieldbus approach was an ambitious concept: a step toward decentralization, including the preprocessing of data in the field devices, which both increases the quality of process control and reduces the computing burden for the centralized controllers [9]. Along with it came the possibility to configure and parameterize the field devices remotely via the bus. This advanced concept, on the other hand, demanded increased communication between the devices that goes far beyond a simple data exchange. This seemed infeasible

to many developers, and still in the mid-1980s, one could read statements like [10]: "The idea of the fieldbus concept seems promising. However, with reasonable effort it is not realizable at present."

The alternative and somewhat more conservative approach was the development of so-called "field multiplexers," devices that collect process signals in the field, serialize them, and transfer them via one single cable to a remote location where a corresponding device demultiplexes them again [11]. For quite some time, the two concepts competed and coexisted [12], but ultimately the field multiplexers mostly disappeared, except for niches in process automation, where many users still prefer such "Remote I/O" systems despite the advantages of fieldbus solutions [13]. The central field multiplexer concept of sampling I/O points and transferring their values in simple data frames also survived in some fieldbus protocols especially designed for low-level applications.

The desire to cope with the wiring problem getting out of hand in large installations was certainly the main impetus for the development of fieldbus systems. Other obvious and appealing advantages of the concept are modularity, the possibility to easily extend installations, and the possibility to have much more intelligent field devices that can communicate not just for the sake of process data transfer, but also for maintenance and configuration purposes [14,15]. A somewhat different viewpoint that led to different design approaches was to regard bus systems in process control as the spine of distributed real-time systems [16]. While the wiring optimization concepts were in many cases rather simple bottom-up approaches, these distributed real-time ideas resulted in sophisticated and usually well investigated top-down designs.

An important role in the fieldbus evolution has been played by the so-called automation pyramid. This hierarchical model was defined to structure the information flow required for factory and process automation. The idea was to create a transparent, multilevel network—the basis for computer-integrated manufacturing (CIM). The numbers vary, but typically this model comprised up to five levels [7,8]. While the networks for the upper levels already existed by the time the pyramid was defined, the field level was still governed by point-to-point connections. Fieldbus systems were therefore developed also with the aim of finally bridging this gap. The actual integration of field-level networks into the rest of the hierarchy was in fact considered in early standardization [4], for most of the proprietary developments, however, it was never the primary intention.

In the automation pyramid, fieldbusses actually populate two levels: the field level and the cell/process level. For this reason, they are sometimes further differentiated into two classes:

- Sensor–actuator busses or device busses have very limited capabilities and serve to connect very simple devices with, e.g., programmable logic controllers (PLCs). They can be found exclusively on the field level.
- Fieldbusses connect control equipment like PLCs and PCs as well as more intelligent devices. They are found on the cell level and are closer to computer networks.

Depending on the point of view, there may even be a third sublevel [17]. This distinction may seem reasonable but is in fact problematic. There are only few fieldbus systems that can immediately be allocated to one of the groups, most of them are being used in both levels. Therefore, it should be preferable to abandon this arbitrary differentiation.

How do fieldbus systems compare to computer networks? The classical distinction of the different network types used in the automation pyramid hinges on the distances the networks span. From top down, the hierarchy starts with global area networks (GANs), which cover long, preferably intercontinental distances and nowadays mostly use satellite links. On the second level are wide area networks (WANs). They are commonly associated with telephone networks (no matter if analog or digital). Next come the well-known LANs, with Ethernet as the today most widely used specimen. They are the classical networks for office automation and cover only short distances. The highest level of the model shown in Figure 20.1 is beyond the scope of the original definition, but is gaining importance

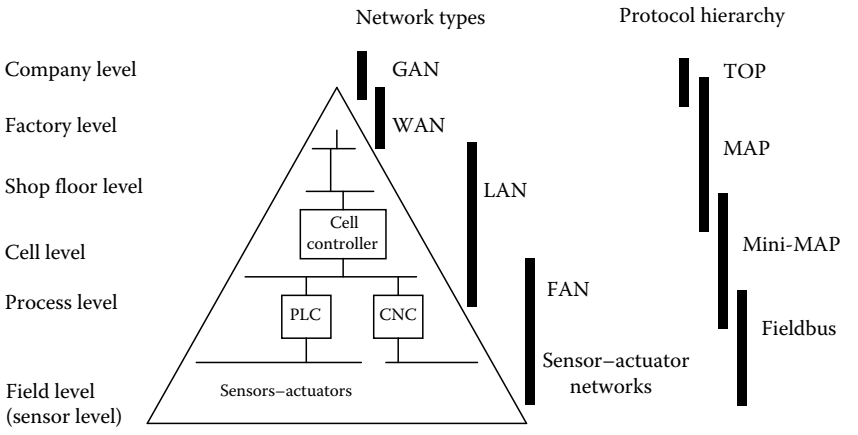


FIGURE 20.1 Hierarchical network levels in automation and protocols originally devised for them. (From Sauter, T., in *The Industrial Communication Handbook*, CRC Press, Boca Raton, FL, 2005, 7.1–7.39. With permission.)

with the availability of the Internet. In fact, Internet technology is penetrating all levels of this pyramid all the way down to the process level.

From GANs to LANs, the classification according to the spatial extension is evident. One step below, on the field level, this criterion fails, because fieldbus systems or field area networks (FANs) can cover even larger distances than LANs. Yet, as LANs and FANs evolved nearly in parallel, some clear distinction between the two network types seemed necessary. As length is inappropriate, the classical borderline drawn between LANs and FANs relies mostly on the characteristics of the data transported over these networks. LANs have high data rates and carry large amounts of data in large packets. Timeliness is not a primary concern, and real-time behavior is not required. Fieldbus systems, by contrast, have low data rates. As they transport mainly process data, the size of the data packets is small, and real-time capabilities are important.

For some time, these distinction criteria between LANs and FANs were sufficient and fairly described the actual situation. Recently, however, drawing the line according to data rates and packet sizes is no longer applicable. In fact, the boundaries between LANs and fieldbus systems have faded. Today, there are fieldbus systems with data rates well above 10 Mbit/s, which is still standard in older LAN installations. In addition, more and more applications require the transmission of video or voice data, which results in large data packets.

On the other hand, Ethernet as the LAN technology is becoming more and more popular in automation and is bound to replace some of today's widely used mid-level fieldbus systems. The real-time extensions being under development tackle its most important drawback and will ultimately permit the use of Ethernet also in low-level control applications. At least for the next 5 years, however, it seems that industrial Ethernet will not make the lower-level fieldbusses fully obsolete. They are much better optimized for their specific automation tasks than the general-purpose network Ethernet. But the growing use of Ethernet results in a reduction of the levels in the automation hierarchy. Hence, the pyramid gradually turns into a flat structure with at most three, maybe even only two levels.

Consequently, a more appropriate distinction between LANs and FANs should be based on the functionality and the application area of these networks. According to this pragmatic argumentation, a fieldbus is simply a network used in automation, irrespective of topology, data rates, protocols, or real-time requirements. Consequently, it need not be confined to the classical field level, it can be found on higher levels (provided they still exist) as well. A LAN, on the other hand, belongs to the office area. This definition is loose, but mirrors the actual situation. Only one thing seems strange at

first. Following this definition, industrial Ethernet changes into a fieldbus, even though many people are inclined to associate it with LANs. However, this is just another evidence that the boundaries between LANs and FANs are fading.

20.3 History

The question of what constitutes a fieldbus is closely linked to the evolution of these industrial networks. The best approach to understanding the essence of the concepts is to review the history and the intentions of the developers. This review will also falsify one of the common errors frequently purported by marketing divisions of automation vendors: that fieldbus systems were a revolutionary invention. They may have revolutionized automation, there is hardly any doubt about it. However, they themselves were only a straightforward evolution that built on preexisting ideas and concepts.

20.3.1 Roots of Industrial Networks

Although the term "fieldbus" appeared only about 25 years ago, the basic idea of field-level networks is much older. Still, the roots of modern fieldbus technology are mixed. Both classical electrical engineering and computer science have contributed their share to the evolution, and we can identify three major sources of influence:

- Communication engineering with large-scale telephone networks
- Instrumentation and measurement systems with parallel busses and real-time requirements
- Computer science with the introduction of high-level protocol design

This early stage is depicted in Figure 20.2. One foundation of automation data transfer has to be seen in the classic telex networks, and also in standards for data transmission over telephone lines. Large distances called for serial data transmission, and many of these comparatively early standards still exist, like V.21 (data transmission over telephone lines) and X.21 (data transmission over special data lines). Various protocols have been defined, mostly described in state machine diagrams and

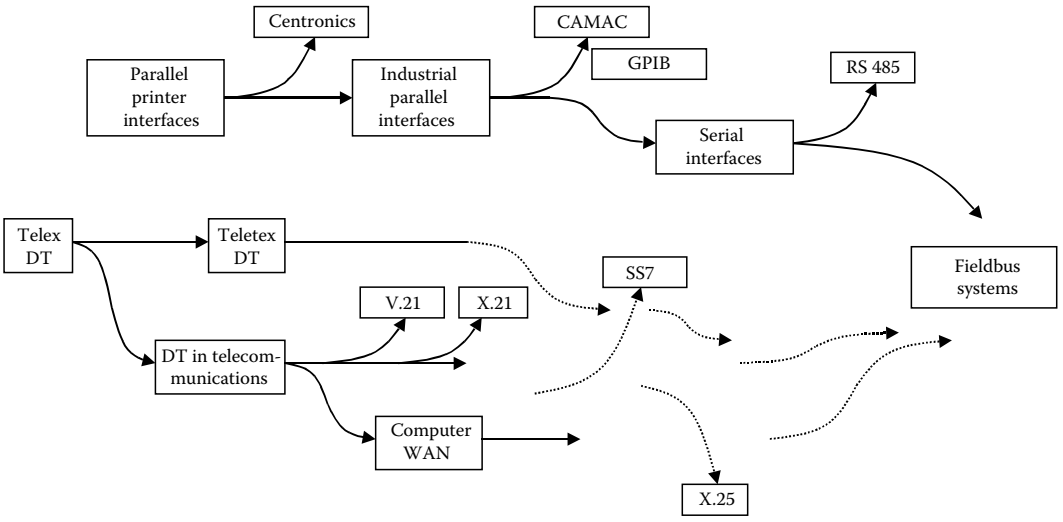


FIGURE 20.2 Historical roots of fieldbus systems. (From Sauter, T., in *The Industrial Communication Handbook*, CRC Press, Boca Raton, FL, 2005, 7.1–7.39. With permission.)

rather simple because of the limited computing power of the devices available at that time. Of course, these communication systems have a point-to-point nature and therefore lack the multidrop characteristic of modern fieldbus systems, but nevertheless they were the origin of serial data transmission. Talking about serial data communication, one should notice that the engineers who defined the first protocols often had a different understanding of the expressions "serial" and "parallel" than we have today. For example, the serial Interface V.24 transmits the application data serially, but the control data in a parallel way over separate control lines.

In parallel to the development of data transmission in the telecommunication sector, hardware engineers defined interfaces for standalone computer systems to connect peripheral devices such as printers. The basic idea of having standardized interfaces for external devices was soon extended to process control and instrumentation equipment. The particular problems to be solved were the synchronization of spatially distributed measurement devices and the collection of measurement data from multiple devices in large-scale experimental setups. This led to the development of standards like computer automated measurement and control (CAMAC, mainly used in nuclear science) and general purpose interface bus (GPIB, later also known as IEEE 488). To account for the limited data-processing speed and real-time requirements for synchronization, these bus systems had parallel data and control lines, which is also not characteristic for fieldbus systems. However, they were using the typical multidrop structure.

Later on, with higher integration density of integrated circuits and thus increased functionality and processing capability of microcontrollers, devices became smaller and portable. The connectors of parallel bus systems were now too big and clumsy, and alternatives were sought [18]. The underlying idea of developments like I^2C [19] was to extend the already existing serial point-to-point connections of computer peripherals (based on the RS 232) to support longer distances and finally also multidrop arrangements. The capability of having a bus structure with more than just two connections together with an increased noise immunity due to differential signal coding eventually made the RS 485 a cornerstone of fieldbus technology up to the present day.

Historically the youngest root of fieldbus systems, but certainly the one that left the deepest mark was the influence of computer science. Its actual contribution was a structured approach to the design of high-level communication systems, contrary to the mostly monolithic design approaches that had been sufficient until then. This change in methodology had been necessitated by the growing number of computers used worldwide and the resulting complexity of communication networks. Conventional telephone networks were no longer sufficient to satisfy the interconnection requirements of modern computer systems. As a consequence, the big communication backbones of the national telephone companies gradually changed from analog to digital systems. This opened the possibility to transfer large amounts of data from one point to another. Together with an improved physical layer, the first really powerful data transmission protocols for WANs were defined, such as X.25 (packet switching) or SS7 (common channel signaling). In parallel to this evolution on the telecommunications sector, LANs were devised for the local interconnection of computers, which soon led to a multitude of solutions. It took nearly a decade until Ethernet and TCP/IP (transmission control protocol/Internet protocol) finally gained the dominating position they have today.

20.3.2 Evolution of Fieldbusses

The preceding section gave only a very superficial overview of the roots of networking, which not only laid the foundations of modern computer networks, but also of those on the field level. But let us now look more closely at the actual evolution of the fieldbus systems. Here again, we have to consider the different influences of computer science and electrical engineering. First and foremost, the key contribution undoubtedly came from the networking of computer systems, when the ISO/OSI (open systems interconnection) model was introduced [20,21]. This seven-layer reference model was (and still is) the starting point for the development of many complex communication protocols.

The first application of the OSI model to the domain of automation was the definition of the manufacturing automation protocol (MAP) in the wake of the CIM idea [22]. MAP was intended to be a framework for the comprehensive control of industrial processes covering all automation levels, and the result of the definition was a powerful and flexible protocol [23]. Its complexity, however, made implementations extremely costly and hardly justifiable for general-purpose use. As a consequence, a tightened version called MiniMAP and using a reduced model based on the OSI layers 1, 2, and 7 was proposed to better address the problems of the lower automation layers [24]. Unfortunately, it did not have the anticipated success either. What did have success was the manufacturing message specification (MMS). It defined the cooperation of various automation components by means of abstract objects and services and was later used as a starting point for many other fieldbus definitions [25]. The missing acceptance of MiniMAP as well as the inapplicability of the original MAP/MMS standard to time-critical systems [26] was finally the reason for the IEC to launch the development of a fieldbus based on the MiniMAP model, but tailored to the needs of the field level. According to the original objectives, the higher levels of the automation hierarchy should be covered by MAP or PROWAY [22].

Independent of this development in computer science, the progress in microelectronics brought forward many different integrated controllers, and new interfaces were needed to interconnect the ICs in an efficient and cheap way. The driving force was the reduction of both the interconnect wires on the printed circuit boards and the number of package pins on the ICs. Consequently, electrical engineers—without the knowledge of the ISO/OSI model or similar architectures—defined simple busses like the I^2C. Being interfaces rather than full-fledged bus systems, they have very simple protocols, but they were and still are widely used in various electronic devices.

Long before the "invention" of board-level busses, the demand for a reduction of cabling weight in avionics and space technology had led to the development of the MIL STD 1553 bus, which can be regarded as the first "real" fieldbus. Introduced in 1970, it showed many characteristic properties of modern fieldbus systems: serial transmission of control and data information over the same line, master/slave structure, the possibility to cover longer distances, integrated controllers, and it is still used today. Later on, similar thoughts (reduction of cabling weight and costs) resulted in the development of several bus systems in the automotive industry, but also in the automation area. A characteristic property of these fieldbusses is that they were defined in the spirit of classical interfaces, with a focus on the lower two protocol layers, and no or nearly no application layer definitions. With time, these definitions were added to make the system applicable to other areas as well. CAN is a good example of this evolution: For the originally targeted automotive market, the definition of the lowest two OSI layers was sufficient. Even today, automotive applications of CAN typically use only these low-level communication features because they are easy to use and the in-vehicle networks are usually closed. For applications in industrial automation, however, where extensibility and interoperability is an important issue, higher-level functions are important. So, when CAN was found to be interesting also for other application domains, a special application layer was added. The lack of such a layer in the original definition is the reason why there are many different fieldbus systems (like CANopen, smart distributed system [SDS], DeviceNet) using CAN as a low-level interface.

From today's point of view, it can be stated that all fieldbusses that still have some relevance were developed using the "top-down" or "computer science-driven" approach, i.e., a proper protocol design with abstract high-level programming interfaces to facilitate usage and integration in complex systems. The fieldbusses that followed the "bottom-up" or "electrical engineering-driven" approach, i.e., that were understood as low-level computer interface, did not survive due to their inflexibility and incompatibility with modern software engineering, unless some application layer functions were included in the course of the evolution.

From the early 1980s on, when automation made a great leap forward with PLCs and more intelligent sensors and actuators, something like a gold rush set in. The increasing number of devices used in many application areas called for a reduced cabling, and microelectronics had grown mature

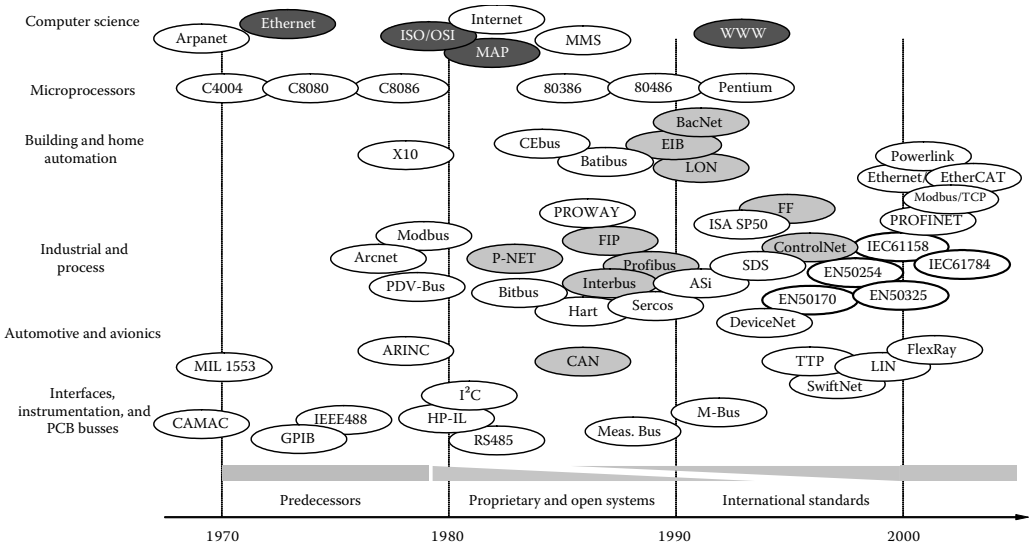


FIGURE 20.3 Milestones of fieldbus evolution and related fields.

enough to support the development of elaborated communication protocols. This was also the birth date for the fieldbus as an individual term. Different application requirements generated different solutions, and from today's point of view it seems that creating new fieldbus systems was a trendy and fashionable occupation for many companies in the automation business. Those mostly proprietary concepts never had a real future, because the number of produced nodes could never justify the development and maintenance costs. Figure 20.3 depicts the evolution timeline of fieldbus systems and their environment. The list of examples is of course not comprehensive, only systems that still have some significance have been selected. Details about the individual solutions are summarized in the tables in the appendix.

As the development of fieldbus systems was a typical "technology push" activity driven by the device vendors, the users first had to be convinced of the new concepts. Even though the benefits were quite obvious, the overwhelming number of different systems appalled rather than attracted the customers, who were used to perfectly compatible current-loop or simple digital I/Os as interfaces between field devices and controllers and were reluctant to use new concepts that would bind them to one vendor. What followed was a fierce selection process where not always the fittest survived, but often those with the highest marketing power behind them. Consequently, most of the newly developed systems vanished or remained restricted to small niches. After a few years of struggle and confusion on the user's side, it became apparent that proprietary fieldbus systems would always have only limited success and that more benefit lies in creating "open" specifications, so that different vendors may produce compatible devices, which gives the customer back their freedom of choice [8,27]. As a consequence, user organizations were founded to carry on the definition and promotion of the fieldbus systems independent of individual companies [28]. It was this idea of open systems that finally paved the way for the breakthrough of the fieldbus concept.

The final step to establish the fieldbus in the automation world was international standardization. The basic idea behind it is that a standard establishes a specification in a very rigid and formal way, ruling out the possibility of quick changes. This attaches a notion of reliability and stability to the specification, which in turn secures the trust of the customers and, consequently, also the market position. Furthermore, a standard is vendor-independent, which guarantees openness. Finally, in many countries standards have a legally binding position, which means that when a standard can be

applied (e.g., in connection with a public tender), it has to be applied. Hence, a standardized system gains a competitive edge over its nonstandardized rivals. This position is typical for, e.g., Europe (see Ref. [29] for an interesting U.S.-centric comment). It is therefore no wonder that after the race for fieldbus developments, a race for standardization was launched. This was quite easy on a national level, and most of today's relevant fieldbus systems soon became national standards. Troubles started when international solutions were sought. The fieldbus standardization project of IEC, which was started in the technical subcommittee SC65C in 1985, had the ambitious objective of creating one single, universally accepted fieldbus standard for factory and process automation [30,69]. Against the backdrop of a quickly evolving market and after 14 years of fierce technical and increasingly political struggles, this goal was abandoned with the multi-protocol standards IEC 61158 and IEC 61784-1 [31,32,138]. In other application domains, other standards were defined, so that the fieldbus world today consists of a sumptuous collection of well-established approaches.

20.4 Communication Fundamentals—The OSI Model

It has been stated before that the definition of the OSI model was an essential cornerstone for the development of fieldbus systems. The attempts at creating a reference model for data communication arose from the fact that at the time, there were a series of computer networks all of which were incompatible with one another. The expansion of these networks was therefore limited to a specific circle of users. Data transmission from one network to another was only possible with great investment in specialized hardware and software solutions. It was the aim of the OSI model to counteract this development. ISO introduced the concept of an "open system." Such systems consist of hardware and software components that comply with a given set of standards. These standards guarantee that systems from different manufacturers are compatible with one another and can easily communicate.

To alleviate handling the rather complex task of data communication, it was decided in the committee to partition it into a strictly hierarchical, layered model. All relevant communication functions were counted up and ordered into overlying groups building on one another. On that basis, a reasonable degree of modularization was sought, and seven layers seemed a feasible compromise. In fact, there is no other significance or mystery in the number of layers. The great significance of the OSI model and its value for practical use came about due to the consistent implementation of three essential concepts:

- *Protocol*. The term protocol denotes a set of rules that govern the communication of layers on the same level. If layer N of open system 1 wishes to contact layer N of open system 2, both systems must adhere to specific rules and conventions. Together, these rules make up the protocol of layer N. Layers lying on the same level are also called peer layers.
- *Service*. This represents any service made available by one layer (called "service provider") to the layer directly above it (called "service user"). It is important to notice that for the services of a layer, the OSI model only defines their functionality and not how they are actually implemented.
- *Interface*. There is an interface between every two layers. This clearly defined interface specifies which services are offered by the lower layer to the upper layer, i.e., how the service user can access the services of the service provider, what parameters need to be transferred and what the expected results are.

It is a common misconception to think that the OSI model describes or prescribes actual implementations. It only establishes the effects of each of the layers. Suitable standards were additionally

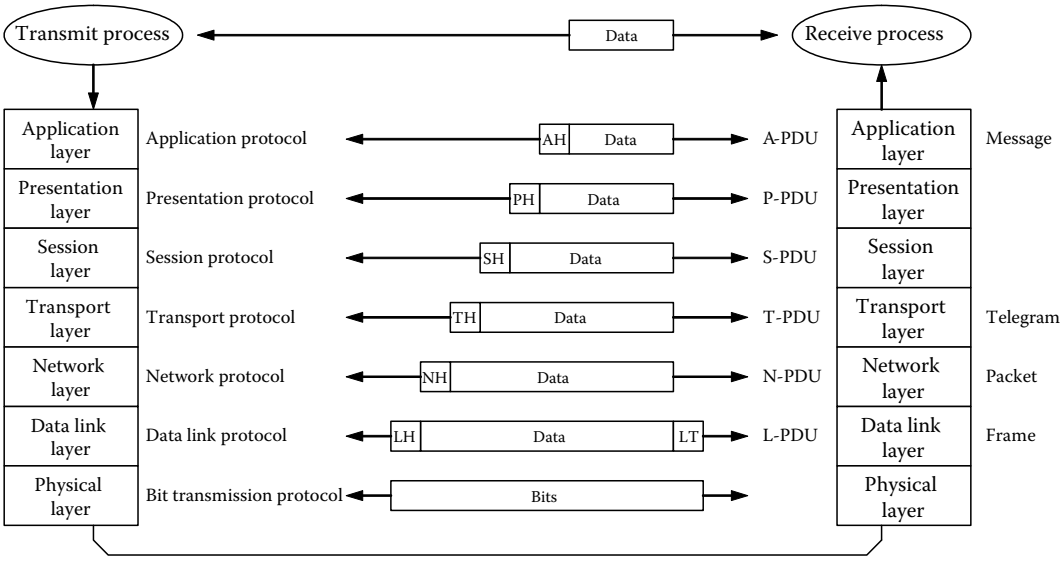


FIGURE 20.4 Layer structure of the OSI model and data frame formation.

worked out for the layers. These standards are not part of the reference model and were later published in their own right.

20.4.1 Layer Structure

The clear distinction between horizontal and vertical communication was the key to the definition of interconnectable systems. Figure 20.4 shows the layer structure and how the data are passed from one process to another. A system application initiates a transmission via layer 7 of its communication stack. To fulfill this task, layer 7 prepares data for its peer layer on the receiving side, packages this and requests services of layer 6 to transmit this data. Layer 6 does the same. It prepares data for the peer layer 6′ to fulfill its task and requests services from the next lower layer 5, and so on all the way down to layer 1, which actually transmits the data. On the way down the layers, the data of the application process are augmented by layer-specific data needed to execute the respective protocols. These data are typically address and control information that is mostly combined in a protocol header. In addition, the data may be segmented into individual packets to match the allowed maximum packet size for a given layer. This way, the number of bits being actually transmitted can be significantly larger than the pure user data provided by the application process, and the communication overhead can be substantial. On the receiving side, the peer layers strip all this additional information to recover the user data for the application process. The basic functions of the individual layers are briefly described below.

Layer 1 (Physical Layer)
As the name suggests, this layer describes all mechanical, physical, optical, electrical, and logical properties of the communication system. This includes, e.g., definition of the connector type, impedances, transmission frequencies, admissible line lengths, line types, the type of coding of individual bits (such as NRZI: Non-Return to Zero Inverted, RZ: Return to Zero, Manchester Coding), simultaneous transmission of energy and data, signal energy imitations for intrinsically safe applications, and the like. One of the most widely used standards for layer 1 in fieldbus technology is the

RS 485 interface, which naturally only forms a small part of a complete layer 1 definition. Essentially, layer 1 presents to its upper layer all that is needed to transfer a given data frame.

Layer 2 (Data Link Layer)
This layer is a pure point-to-point connection with the task of guaranteeing transmission between two network nodes. This firstly involves the formation of the data frame (Figure 20.4), which typically contains a header with control and address information and the actual data. The second task of the data link layer is the coding and checking of the frame (e.g., via CRC: cyclic redundancy check), to allow transmission errors to be detected or even corrected. This also includes the checking of timeouts, or verification that the corresponding responses and confirmations are received from the opposite side. The data link layer thus provides the following service to the above lying layer 3: Setup of a logic channel to an opposite end device without intermediate nodes, or the transmission of a data frame between two end points. In practice, layer 2 becomes too overloaded with functions to allow for a straightforward implementation. Therefore, it is usually subdivided into the logical link control (see IEEE 802.2), which sets up the connection to layer 3 (to which the error detection mechanism is assigned) and the medium access control (MAC) to link to layer 1 (this generally controls who is able to transmit when).

Layer 3 (Network Layer)
If there are nodes between the end points of an end-to-end connection, packets must be routed. In layer 3, the paths between origin and destination are established via the specified target addresses. This is easy if the corresponding path lists are available in the nodes. It becomes more complicated when the paths are to be optimized on the basis of various criteria such as cost, quality, load, delay times, etc., if the path conditions change during a transmission, packets need to take different paths due to bandwidth considerations, or the packet size is unsuitable for certain paths. The task of layer 3 is by no means a trivial one, especially when there are various physical transmission media within the network with different transmission speeds. It is also necessary to ensure that congestion does not occur along the paths, which would cause the maximum delay times to be exceeded.

A differentiation is made between connectionless and connection-oriented services. In the case of a connectionless service (datagram service), there is no allocation of fixed channels; every transmitted package must include the complete address and is sent as an independent unit. With a connection-oriented service (virtual circuit service), a virtual channel is made available, which from the point of view of the user offers the advantage that the data packets need not include any addresses. One of the first protocols of this type to be implemented and which is still in use today is the X.25 (ISO 8473). For fieldbus systems, such virtual circuits (and therefore connection-oriented layer-3 services) do not play any practical role. In any case, layer 3 presents to its upper layer a valid path through the network for one individual data packet.

Layer 4 (Transport Layer)
The transport layer sets up an end-to-end connection. This means that the receiver does not route the data further, but passes them on to layer 5, already prepared. There are various mechanisms available in layer 4. If the data to be transmitted are too big, layer 4 can split them up and transmit each piece individually. If the transmission times are long or there are a number of possible transmission paths, it is useful to number the split packets. The receiver station must then recombine the individual packets in the right order. On layer 4, connection-oriented and connectionless protocols are available as well. The most popular examples are TCP for connection-oriented and user datagram protocol (UDP) for connectionless data transmission. They originated in the Internet world but are becoming increasingly important in the field-level networking domain.

Layer 5 (Session Layer)
The main task of layer 5 is to bring together several end devices into a session and to synchronize the "conversation." This also involves identification or authentication (e.g., password check) and the handling of very large messages. Close cooperation between this layer and the operating system is of vital significance. For this reason, layers 6 and 7 are implemented in fieldbus practice with high transparency for these particular layer-5 functions, or a separate channel is created to bypass them. It is also the task of layer 5 to introduce any necessary synchronization markers, so that it knows when to resume after a breakdown in communication.

Layer 6 (Presentation Layer)
The presentation layer interprets the incoming data and codes the data to be transmitted. This means that level 6 carries out syntactic and semantic tasks. These include, e.g., the meaning of the sequence of bits of a character, to be interpreted as a letter, interpretation of currency as well as physical units, cryptographic tasks, etc. For encoding, ISO defined the standards Abstract Syntax Notation 1 or Basic Encoding Rules which are frequently used.

Layer 7 (Application Layer)
Layer 7 is a boundary layer (interface to the application) and with that occupies a special position. It forms the interface between the application and the communication unit. In this level, the procedures or protocol processes of various application functions are defined, for calling up data, file transfer, etc. The purpose of layer 7 is a transparent representation of the communication. If, e.g., a system accesses databases via the communication unit, layer 7 must be designed so that it does not require any knowledge of the individual tasks of the underlying layers. With that, an efficient communication system allows a database that is distributed among various different locations to be viewed as a single interconnected database. An intelligent switch that transmits information via an OSI communication unit, such as "turn light on," does not need to know anything about the actual communication protocol; it simply knows the "name" of the lights as well as the functions "turn light on," "turn light off, " or "dim light by 40%," etc.

Based on these definitions, it is possible to set up complex communication systems in a structured way. Moreover, the strictly hierarchical layout of the model allows interconnection of heterogeneous systems on different layers. Through the use of repeaters, one can overcome the limitations of a given physical layer. The interconnecting device shares a common data link layer. Bridges interconnect different networks by translating data and protocols on layer 3. Routers link networks on layer 4, whereas gateways (or more precisely, application layer gateways) interconnect entirely different communication systems on the application layer.

20.4.2 Communication Services

Figure 20.5 shows in detail how data are exchanged between service user and service provider or between two peer layers. The interface between two neighboring layers is called service access point (SAP). The vertical and horizontal communication is made up of two important units:

- *Service Data Unit* (SDU). Communication of layer N + 1 with the underlying layer N occurs via its services, or more accurately via the interfaces of these services. For layer N, the transferred data represent pure user data that are passed on to the next lowest layer for further processing.
- *Protocol Data Unit* (PDU). Communication between two peer layers is implemented via so-called PDUs. These represent the core element of the rule set that can only be understood and correctly interpreted by the peer layers. A PDU consists of the

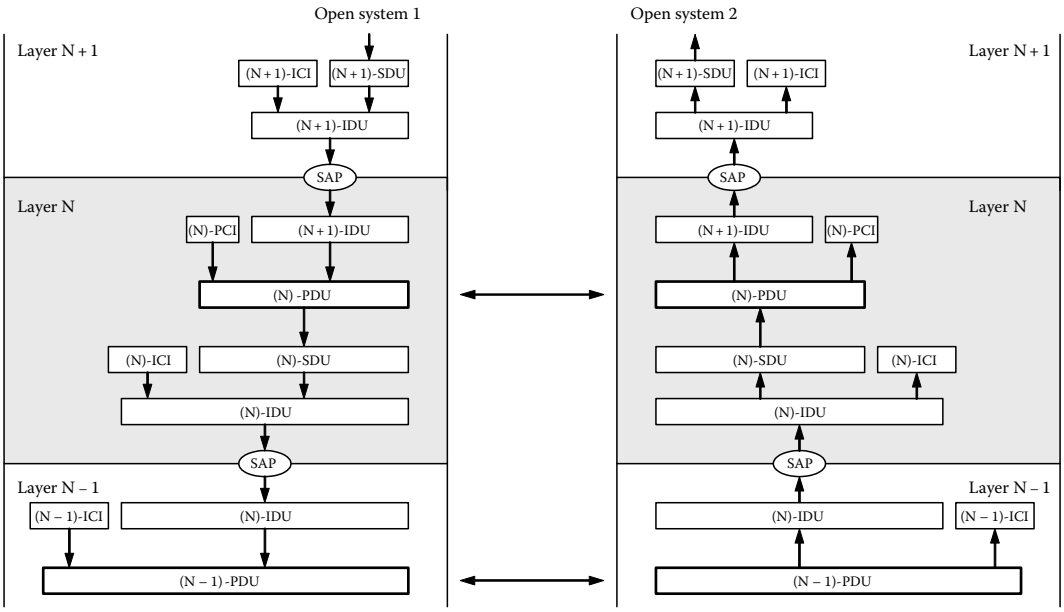


FIGURE 20.5 Communication in the OSI model.

transmitted user data supplemented with parameters of the interface (interface control information [ICI]) and unique control information (protocol control information [PCI]).

The exact procedure is highlighted in Figure 20.5. The user data of layer N that are to be transmitted to its peer layer are put into an SDU. The interface process of layer N adds an ICI to the message header (the SDU header), whereby the ICI and SDU together form the interface data unit (IDU). The IDU is then transferred via the SAP. In the underlying layer, the ICI is now decoded and a corresponding process is initiated. The following process (execution of a protocol) should be viewed as a logic process, except in the lowest layer, the physical layer. In the procedure initiated by the ICI, layer N–1 packs the SDU into a PDU. This involves adding a PCI to the SDU. The assembled PDU is now transmitted, which means that layer N–1 in turn uses the lower layers and itself now behaves as a service user. Only layer 1 actually physically transmits the information.

In even more detail, the interaction between the individual layers is governed by a series of operations defined in the OSI model. These operations are called service primitives. There are basically four different service primitives: Request, req; indication, ind; confirmation, con; and response, res. Each service primitive that is called up by the service user is termed a request. With the request, layer N receives the order to execute a specific task. The respective task and corresponding data are converted into a corresponding PDU. In accordance with the OSI model, the service provider uses the services of the underlying layer N–1, to carry out its task. This interaction continues until the lowest layer entrusts the data to the physical medium. On the receiver side, the peer layer is activated via an indication (or sometimes even a whole series of indications). After the remote layer N has decoded the PDU and extracted the control information, the user data are passed on to the above lying layer (i.e., the layer directly above or in the case of the application layer, the actual user process) also by means of an indication. The way back from the receiver side is composed from responses generated by the service users and confirmations issued by the service providers. However, services need not always comprise each of these four service primitives. There are many possible and practically used combinations with varying degree of reliability, as the following examples show:

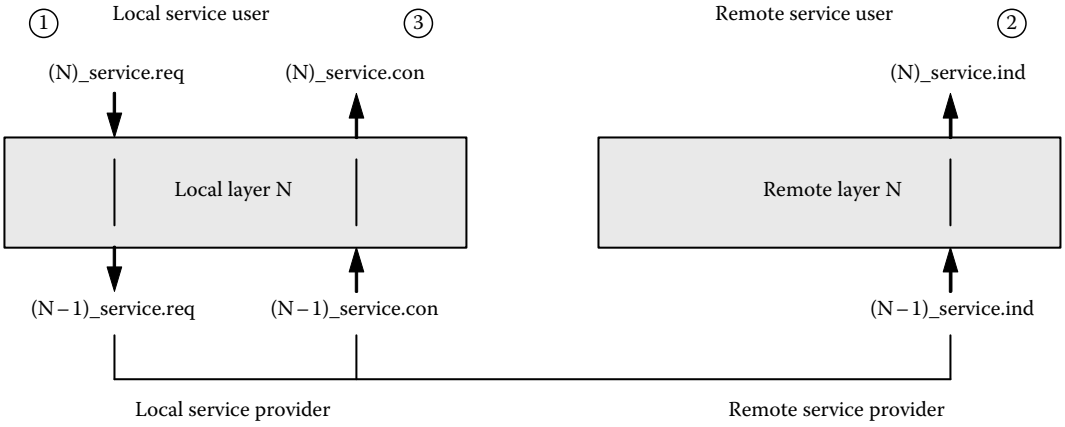


FIGURE 20.6 Locally confirmed service.

1. *Locally confirmed service*. A locally confirmed service (Figure 20.6) comprises a request, an indication, and a confirmation. In this case, layer N of the sender then receives a confirmation from local layer N–1. Based on the parameters transmitted, the service provider can tell whether its underlying layer was able to accept and process the request accordingly. After a brief preprocessing, this local confirmation is also made known to the service user by means of a confirmation service primitive. As the name suggests, a locally confirmed service gives no guarantee that the remote user receives the information transmitted. The confirmation can be omitted at all, which leads to an unconfirmed service.
2. *Confirmed service*. This type of service also consists of a request, an indication and a confirmation. With a confirmed service, however, the peer layer generates an acknowledgment immediately after receiving the indication (Figure 20.7). The acknowledgment is returned to the sender via a service primitive of the remote layer N–1 and signaled to the service provider via an indication of local layer N–1. Note that contrary to the procedure outlined in point 1 above, the transmitted parameters and data come from the partner

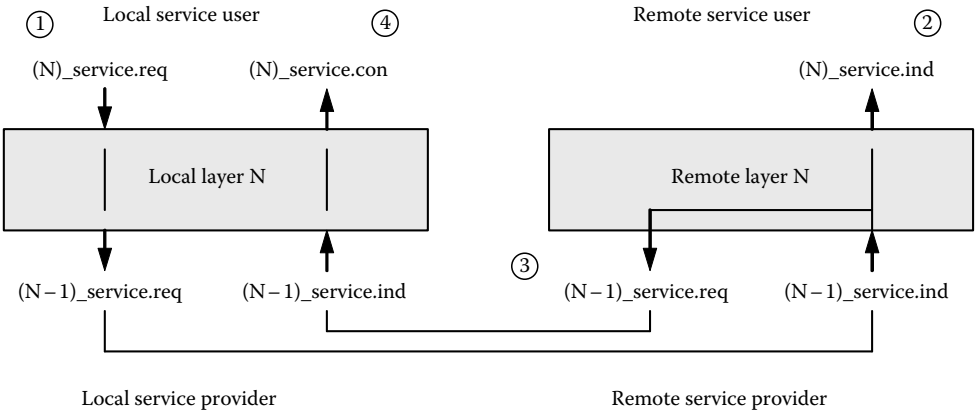


FIGURE 20.7 Confirmed service.

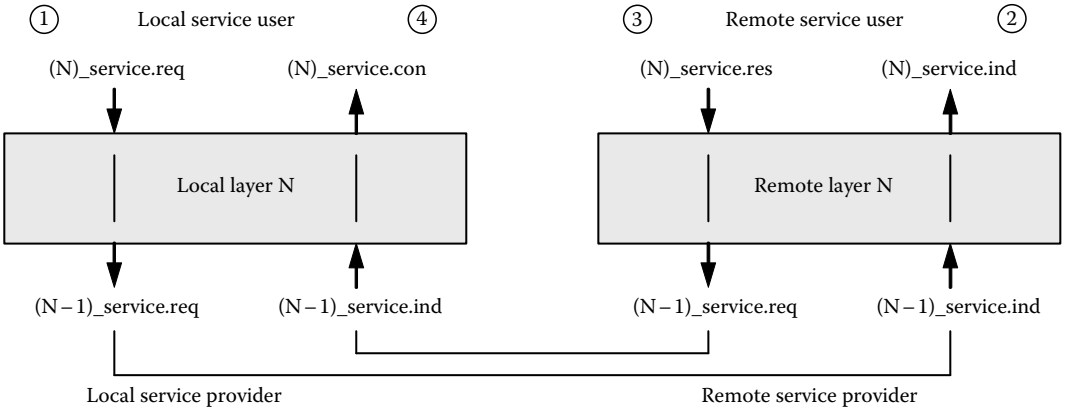


FIGURE 20.8 Answered service.

side. From the indication, layer N can conclude whether or not the service requested by layer N–1 was executed without error. If necessary (i.e., if errors have occurred), this service is used again. This depends on the protocol used in layer N. In all cases, the service user is informed of the output with a suitable confirmation (positive or negative). From this confirmation, it is possible to derive whether or not the originally requested service has satisfactorily been fulfilled by layer N.

3. *Answered service*. This represents the fully fledged use case of all service primitives (Figure 20.8). Here, after an indication on the partner side, a response is generated by the remote service user. The response is transmitted via a service primitive of layer N–1 and passed on as a confirmation to the service user from which the original service request came. This mechanism permits data traffic in both directions. Contrary to the other two services, an answered service always consists of request, indication, response, and confirmation.

In short, request and response are always called up by the service user, the resulting confirmation and indication originate from the corresponding layer. The interaction between the individual layers is best understood if one imagines that the layers lying on top of one another are primarily inactive. The service provider is only "awoken" after a request from the service user. To perform this service, it in turn activates the underlying layer by calling up the appropriate service primitive. This interaction continues down to the lowest layer, which then accesses the transmission medium. On the remote side, the indication of a low lying layer informs the layer lying directly above that a service is to be executed. This indication is then processed in accordance with the service type—it causes either a confirmation or a response.

20.4.3 Quality-of-Service Parameters

Fieldbus systems are to a large extent concerned with process control. The respective data are thus in many cases critical in terms of timing, security, reliability, or the like. This in turn means that the network has to provide services that take this criticality into account. In other words, network services must have a certain quality. The network behavior can therefore be described by so-called quality-of-service (QoS) parameters. The origin of QoS definitions was the usage of the Internet to transport multimedia services, specifically the difficulty of using a packet-switched, statistical multiplexing network to transmit stream-based data that normally require directly switched connections. In automation systems, the situation is quite comparable. Formerly direct connections are

being replaced by networks that influence the quality of control of the implemented application. QoS parameters are a reasonable means to quantify this influence. In principle, QoS refers to the capability to provide resource assurance and service differentiation in a network [33]. In a narrower sense, timeliness and reliability properties are often referred to as network QoS. Some typical QoS parameters relevant for fieldbus systems in an automation context are discussed below.

End-to-End Delay

This is an absolute measure for the total delay a packet experiences on its way from sender to receiver. A requirement on the end-to-end delay D_i of each packet i can be defined such that the delay D_i of each packet i must not exceed a maximum end-to-end delay $D_{\max}$ with a probability greater or equal than $Z_{\min}$,

$$P(D_i \leq D_{\max}) \geq Z_{\min} \tag{20.1}$$

This statistical bound on the end-to-end delay can be easily converted into a deterministic bound by defining $Z_{\min} = 1$.

Delay Jitter

The delay jitter J_k is the variance of the end-to-end delay D_k and is a measure for the stochastic part of the delay. It stems from variations in data-processing times on the end and intermediate nodes a packet visits along its way. I may also be influenced by the mechanism used to grant a node access to the transmission medium or to route traffic in intermediate nodes. Hence, it heavily depends on the overall traffic situation in the network. The jitter is frequently defined by decomposing the network delay

$$D_k = D_{\min} + J_k \tag{20.2}$$

into a deterministic part $D_{\min}$ and a stochastic part J_k. Applications may want to bound the delay jitter J_i to a maximum value $J_{\max}$ with a probability equal or greater than $U_{\min}$,

$$P(J_i \leq J_{\max}) \geq U_{\min} \tag{20.3}$$

Applications requiring jitter bounds are typically highly dynamical control systems needing truly equidistant sampling and execution of commands, such as electric drive controls.

Lifetime

This parameter indicates how long a given data value is meaningful for the application process. It is connected to the network delay and depends on the type of data under consideration. For control operations, there is no point in trying to recover lost or delayed samples of state variables if they are too old. On the other hand, configuration or management data may be valid for a long time. In packet-switched networks, the time-to-live parameter indicates the point in time when a packet becomes obsolete and may be deleted from the network by any node to reduce the network load.

Frequency

The generation rate of the data sources is an important parameter for the planning of the network traffic and resource allocation. If the frequency is too high and the overall data generation exceeds the network capacity, data loss is unavoidable. If the frequency only temporarily is above the capacity, backlog will likely occur, and the delay for individual packets will increase.

Loss Rate

The drop probabilities for different packets on the individual hops can be subsidized by a global "loss probability" or average "loss rate." The most common reason for high loss rates is again network

congestion. The probability of complement can be seen as the reliability of the network path. The reliability defines how reliable packet transmission should be on the network. A requirement can thus be that

$$P(\text{packet received}) \geq W_{\min} \tag{20.4}$$

with $W_{\min}$ as the minimum probability for packet reception.

Packet Ordering
Some services rely on order preservation of a sequence of packets. Obviously, the order can change only if the network topology provides different parallel paths from sender to receiver. In fieldbus systems, this is typically not an issue; however, it can be relevant for general-purpose IP-based networks used in an automation context.

A different possibility to assess the performance of a network is to look at it from the application point of view. Depending on the application's needs, the network must meet certain requirements. These requirements may of course differ for individual data types. With respect to the packet delay as the most important parameter, it has become customary to distinguish several "performance classes" that can be used to define characteristic features for the delay density $f(D)$.

Hard Real-Time
If an application has hard real-time requirements, any given deadline must be met under all circumstances. Missing a deadline will inevitably result in a catastrophic failure of the system. Although the definition of what actually constitutes a catastrophic failure is not always clear, there is at least a concise requirement for the delay, which must be upper-bounded, i.e.,

$$f(D) \equiv 0 \quad \text{for} \quad D \geq D_{\max} \tag{20.5}$$

Soft Real-Time
If missing of a deadline does not lead to a catastrophic failure, the application imposes only soft real-time requirements. This definition is rather loose and can be further split in two subsets:

Guaranteed Delivery
For this requirement, the actual delay of a packet is irrelevant. What counts is that the packet is definitely received by the addressed node. There are no particular constraints for the delay density, it may not even be bounded (if it was, we had effectively a hard real-time situation).

Best Effort
This scenario is the weakest one. Here, the network only attempts to deliver the packets as fast as possible, without giving guarantees concerning deadlines or successful delivery at all. Formally, this comes down to treating the delay density $f(D)$ as conditional distribution,

$$f(D|X) = f(D)p(X) \tag{20.6}$$

where X denotes the successful delivery of the packet, which is independent of the delay distribution. The conditional distribution has the convenient property

$$\int_{-\infty}^{\infty} f(D|X)\mathrm{d}D = p(X = \text{true}) < 1 \tag{20.7}$$

so that the residual probability that a packet is lost in the best effort scenario is implicitly included.

Application Class	RT	GD	BE
Monitoring	+	+	±
Open-loop control	±	++	+
Closed-loop control	+	−	+
File transfer	±	++	−

FIGURE 20.9 Application classes and possible requirements (RT, hard real-time; GD, guaranteed delivery; BE, best-effort) imposed on the network. (From Sauter, T., in *The Industrial Communication Handbook*, CRC Press, Boca Raton, FL, 2005, 7.1–7.39. With permission.)

Distributed applications implemented on automation networks typically cope with process data acquisition, processing, and signaling. They can be categorized into several "application classes," which can be mapped onto the performance classes introduced before (Figure 20.9).

Monitoring/Logging
This application class allows to view or record data within a control application. Typically, selected process parameters are sent to a monitoring station, where they can be tracked or logged. The QoS requirements can range from hard real-time to no requirements at all (meaning an entirely unreliable network is acceptable as well). Examples are remote power meters, weather sensors, or occupancy detectors.

Open-Loop Control
Open-loop control appears where the process characteristics and disturbances are well-known so that feedback paths are not required. The network should provide at least reliable data transfer, but the timing requirements can range from non-real-time to hard real-time. Multidimensional open-loop controls can pose additional timing requirements originating from the time dependencies of the output values. Control data are typically sent periodically, in which case the overhead for providing guaranteed delivery, e.g., by packet retransmission, can be avoided. Examples are light switches, intelligent power outlets, or dot matrix displays.

Closed-Loop Control
Closed-loop control applications are commonly found in automation tasks, where control quality and error compensation are required. The controller does not need to be part of the application; it can be an external system such as a human operator. Timing variations influence stability and performance of a closed-loop control, so this application class has at least a demand for soft real-time behavior. Again, multidimensional control is possible and can enforce more strict timing requirements. Examples are heating, industrial control, and power management.

File Transfer
All distributed automation systems need in one form or the other the transmission of files or at least larger data blocks instead of process data. The typical application is to configure or parameterize field devices either at system start-up or later during runtime. Sometimes, even executable code is transferred, e.g., to update programs running on controllers. The minimum requirement for the data transmission is reliable and guaranteed data transfer. In rare cases, when the upload of a device configuration is time-critical, also hard real-time requirements may apply.

20.5 Fieldbus Characteristics

The application areas of fieldbus systems are manifold, hence many different solutions have been developed in the past. Nevertheless, there is one characteristic and common starting point for all those efforts. Just like today's embedded system networks, fieldbus systems were always designed for efficiency, with two main aspects:

- Efficiency concerning data transfer, meaning that messages are rather short according to the limited size of process data that must be transmitted at a time.
- Efficiency concerning protocol design and implementation, in the sense that typical field devices do not provide ample computing resources.

These two aspects, together with characteristic application requirements in the individual areas with respect to real-time, topology, and economical constraints, have led to the development of concepts that still are very peculiar of fieldbus systems and present fundamental differences to LANs. In the previous section, the general differences in the context of the OSI model were already sketched. This section will discuss some more peculiarities in detail.

20.5.1 Traffic Characteristics and Requirements

The primary function of a fieldbus is to link sensors, actuators, and control units that are used to control a technical process. Therefore, the consideration of the typical traffic patterns to be expected in a given application domain was in most cases the starting point for the development of a new fieldbus. Indeed, the characteristic properties of the various data types inside a fieldbus system differ strongly according to the processes that must be automated. Application areas like manufacturing, process, and building automation pose different "timing and consistency" requirements that are not even invariant and consistent within the application areas [46].

As regards timing, there are two essential philosophies to look at the technical process in a black-box form and describe (and thus convey within the network) its behavior. One is a state-based approach focusing on the status of the process defined by its internal state variables together with its I/Os. These variables are continuously sampled and transmitted in discrete-time fashion and form the basis for continuous process control and monitoring (like temperature, pressure, etc.).

The other philosophy is an event-based one, i.e., data are transmitted only in case of state changes. This approach is well suited for processes or subprocesses that have a somewhat discrete nature and can be modeled as a state machine. Switches obviously lend themselves to such an interpretation. But also in continuous processes, it may be reasonable to transmit process data only if they exceed certain predefined limits. This naturally requires the implementation of control functions locally in the network nodes rather than in a remote control unit.

As far as consistency is concerned, there are on the one hand process data which are continuously updated and on the other hand parameterization data that are transferred only upon demand. In case of error, the former can easily be reconstructed from historical data via interpolation (or simply be updated by new measurements). The system-wide consistency of configuration data, on the other hand, is an important requirement that cannot be met by mechanisms suitable for process data.

Trying to find an abstract and comprehensive classification scheme for all possible fieldbus traffic patterns, we arrive at the traffic types shown in Figure 20.10 and described below. In theory, data traffic generation is exclusively a question of the application process and should be seen independently of the underlying communication system. In practice, however, the communication protocol and the services provided therein heavily influence the possible traffic types—it is one of the peculiarities of fieldbus systems that the two cannot be properly disentangled.

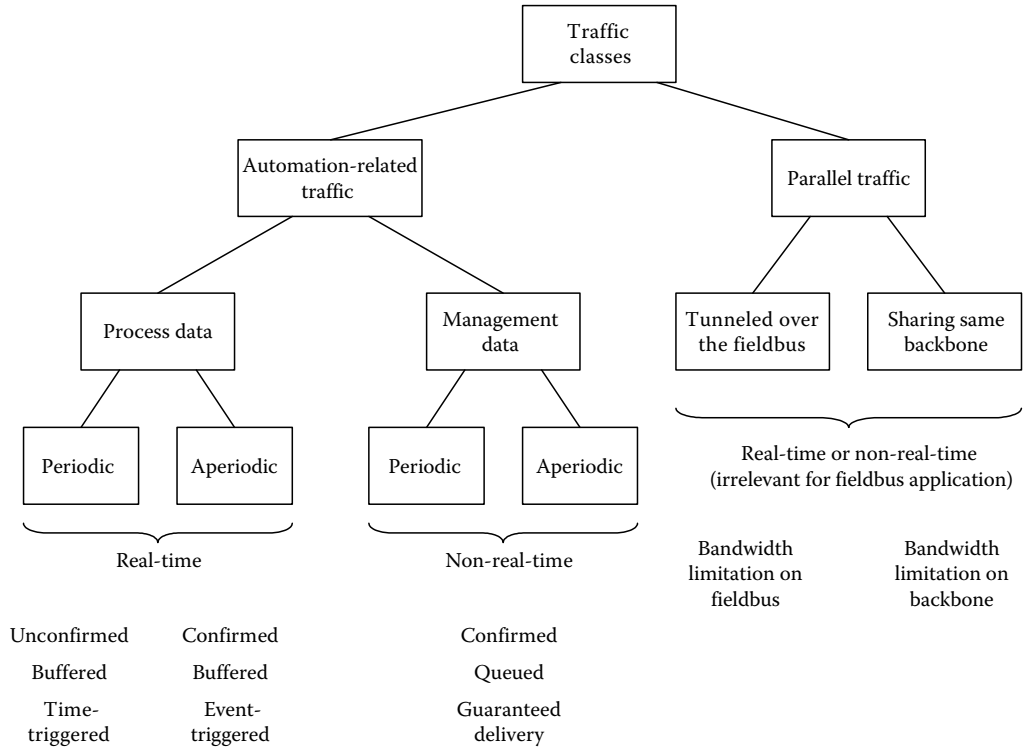


FIGURE 20.10 Traffic classes and typical properties.

20.5.1.1 Process Data

This type is sometimes also called "cyclic or identified" traffic because the communication relations must be known once the application is specified. Depending on the characteristics of the process, such data can be periodic or aperiodic. Periodic traffic mostly relates to the state of a process and is typically handled by some sort of time-slot-based communication strategy, where each variable is assigned a dedicated share of the network bandwidth based on the a priori known sampling time or generation rate of the data source. The frequency or data update rate may also be adaptive, to dynamically change according to the current state of the process (e.g., alarm conditions may require a more frequent sampling of a state variable).

"Aperiodic, acyclic, or spontaneous" traffic is generated on demand in an event-based manner and transmitted according to the availability of free communication bandwidth. If the predominating data transfer scheme is time-slot-based, there may be spare slots or some idle time in the periodic traffic explicitly reserved for this purpose.

From an application point of view, process data typically are real-time data and require timely delivery. For both periodic and aperiodic data, this means that they should be transmitted (and received) within a limited time window to be meaningful for the purpose of process control. For aperiodic data, there is the additional restriction that the service ought to be reliable, i.e., data loss should not occur or at least be detected by some appropriate mechanism.

20.5.1.2 Management Data

This type is also called "parameterization or configuration" data and generally refers to all data that are needed to set up and adjust the operation of the automation system as such. Settings of distributed

application processes (such as control functions localized in the network nodes), communication and network parameters, and more general all network management data belong to this class. Management data typically are aperiodic as they occur infrequently depending on the state of the complete system. Traditionally, they are therefore often transmitted in some dedicated parameter channels that use a small amount of communication bandwidth left over by the (mostly periodic) process data.

In some cases, management data may also be needed periodically or quasi-periodically, e.g., to update session information after a certain time, to exchange authentication information after a given number of messages, or to change communication parameters on a routine basis. Nevertheless, the communication mechanisms for such periodic management messages likely do not differ from the aperiodic ones; they are simply invoked on a periodic basis.

Contrary to process data, management data are mostly not real-time data, which means that timely delivery is not that important. What is more important, however, is guaranteed and correct delivery, which influences the communication services that are used for this type of data. Confirmed services are mandatory.

20.5.1.3 Parallel Traffic

This traffic type does not belong to the applications processes concerned with the actual control of the technical process. Rather, it is generated by independent parallel processes and shares the communication medium. In traditional fieldbus systems, which were closed environments, this type of traffic did not exist. With growing interconnection of automation networks, however, it becomes relevant, either because external traffic (such as IP traffic) may be routed or tunneled through the fieldbus or because fieldbus traffic may be routed through a shared backbone network. In particular, this is an issue for Industrial Ethernet solutions, which in general foresee some sort of general-purpose IP channel for devices not belonging to the automation system.

From the viewpoint of the automation process, care must be taken that the fieldbus traffic is not negatively influenced. This needs to be done by appropriate QoS arrangements that guarantee reasonable bandwidth and latency to the fieldbus application traffic in case a shared backbone is used. In the other case, the fieldbus network management must ensure that the parallel traffic cannot consume an arbitrary amount of communication resources or block other processes.

20.5.1.4 Implications for the Fieldbus

The various traffic types have also implications for the way communication in fieldbus systems is handled, both from a protocol and from an implementation point of view. Data which are exchanged on a cyclic basis are usually sent via connectionless services. The reason behind this is that for periodically updated process data, it makes no sense to require a confirmation. If a message containing process data gets lost, resending it is not sensible because by the time the data might arrive at the receiver, they are outdated, anyway, and new data might already be available. With respect to practical implementation, the handling of such data, especially at the receiver side, is mostly implemented by means of buffers. In these buffers, the latest data always overwrite older values, even when they are basically implemented in a first in, first out (FIFO) structure. In this case, the "FIFO full" signal is suppressed, so that always the most recent values are in the buffer.

On the contrary, acyclic data need special precautions, irrespective of whether they are related to process variables or management data. Loss of such data is not desirable and might be detrimental to the overall application (e.g., if alarm events are not received). Hence, mechanisms involving some sort of acknowledgment must be used to allow for retransmissions in case of data loss. These mechanisms can be implemented on the lower communication layers or the application level, depending on the basic communication services a fieldbus offers. In terms of actual implementation, such messages are typically handled in queues. As opposed to buffers, messages are not overwritten. If the queue is full,

no new messages are accepted until earlier entries have been consumed and deleted. This ensures that no messages are lost before they are processed by the node.

Stimulated by the different ways of looking at and handling data exchange in fieldbus systems (and embedded systems in general), two opposing paradigms have been established in the past and have ignited a long, partly fierce debate: the "time-triggered" and the "event-triggered" paradigm [140]. The time-triggered approach is specifically suited for periodic real-time data, and many fieldbus systems actually use it in one form or another. The event-triggered approach was designed following the idea that only changes in process variables are relevant for transmission. An additional aspect behind it was that such events should be broadcast in the network, so that every node potentially interested in the data can receive them. This makes extension of the network fairly easy by just adding new nodes if the message identifiers they need for their application are known.

20.5.2 Fieldbus Systems and the OSI Model

Like all modern communication systems, fieldbus protocols are essentially modeled according to the ISO/OSI model. However, in most cases only the layers 1, 2, and 7 are actually used [14]. This is in fact a tribute to the lessons learnt from the MAP failure, where it was found that a full seven-layer stack requires far too many resources and does not permit an efficient implementation. For this reason, the MiniMAP approach and based on it the IEC fieldbus standard explicitly prescribes a three-layer structure consisting of physical, data link, and application layer. But what about the other layers and the functions defined therein?

The reduced and simplified protocol stack reflects the actual situation found in many automation applications pretty well, anyway. Many fieldbusses are single-segment networks with limited size, and extensions are realized via repeaters or, at most, bridges. Therefore, network and transport layer—which contain routing functionality and end-to-end control—are simply not necessary. The same applies to the upper layers. Originally, fieldbus systems were not meant to be very sophisticated. Fully implemented session and presentation layers are therefore not needed, either. Still, certain functions from the layers 3–6 might be needed in reduced form. Rudimentary networking aspects could be required or specific coding rules for messages that are better suited for the limited resources available in typical fieldbus nodes. In such cases, these functions are frequently included in the layer 2 or 7. For the IEC 61158 fieldbus standard, the rule is that layer 3 and 4 functions can be placed either in layer 2 or layer 7, whereas layer 5 and 6 functionalities are always covered in layer 7 (Figure 20.11) [45].

It would nevertheless be deceptive to think that all fieldbus systems just consist of a physical, data link, and application layer. There are several examples where other layers were explicitly defined. Particularly in the building automation domain, the situation is different. Owing to the possibly

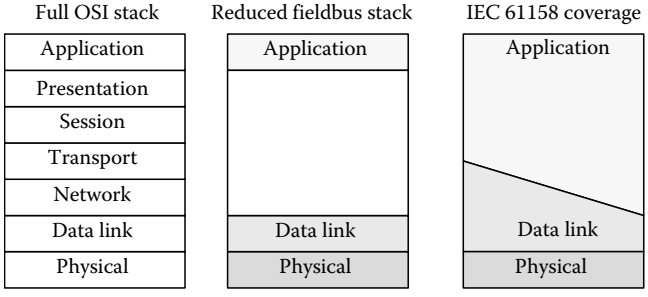


FIGURE 20.11 Layer structure of a typical fieldbus protocol stack as defined by the IEC 61158. (From Sauter, T., in *The Industrial Communication Technology Handbook*, CRC Press, Boca Raton, FL, 2005, 24.1–24.19. With permission.)

high number of nodes, these fieldbus systems must offer the capability of hierarchically structured network topologies, and a reduction to three layers is not sensible. For instance European installation bus (EIB) and Konnex use also the network and transport layers to implement routing through the hierarchical network as well as connection-oriented and connectionless end-to-end communication functions. BACnet uses the network layer as well, which is especially important as BACnet was devised as higher-layer protocol to operate on different lower-layer protocols and links such as Ethernet, MS/TP (master–slave/token passing as inexpensive data-link layer protocol based on the RS 485 standard for the physical layer), and LonTalk. For such a heterogeneous approach, a uniform network layer is essential.

The probably most elaborate protocol structure is the fieldbus world is exhibited by LonWorks. Even though it is today chiefly used in building automation, it was in fact designed as a general-purpose control network (LON stands for local operating network) without any particular application area in mind, hence it resembles much more a LAN than a conventional, highly efficient fieldbus. In the LonTalk protocol, all seven OSI layers are defined, even though layer 6 is rather thin in terms of functionality. Specific characteristics are a rich layer 3 that supports a variety of different addressing schemes and advanced routing capabilities, the support of many different physical layers (a common aspect in all building automation networks), and a large number of various communication objects not just for process data exchange and network management, but also for advanced functions like file transfer.

Among the fieldbus systems mainly used in industrial and process automation, ControlNet and P-NET are particular in that they implement also layers 3 and 4. An outstanding characteristic of P-NET is its capacity for multi-network structures, where so-called multi-port masters can link multiple segments to any arbitrary structure. Layer 3 provides a source routing mechanism (where the path through the network must be defined inside the packet) to manage transmission even in meshed networks. Layer 4 is called service layer and actually contains definitions and processing rules for communication objects that go beyond the usual end-to-end functionality of the OSI transport layer.

An essential part of fieldbus protocol stacks are comprehensive application layers. They are indispensable for open systems and form the basis for interoperability. Powerful application layers offering abstract functionalities to the actual applications, however, require a substantial software implementation effort, which can negatively impact the protocol processing time and also the costs for a fieldbus interface. This is why in some cases (like Interbus, PROFIBUS-DP/PA, or CAN) an application layer was originally omitted. While the application areas were often regarded as limited in the beginning, market pressure and the desire for flexibility finally enforced the addition of higher-layer protocols, and the growing performance of controller hardware facilitated their implementation. CAN is a good example for this because a plethora of protocols (like CANopen, SDS, DeviceNet) appeared in the course of time on the basis of the original CAN layers 1 and 2.

A further vital aspect of any network is an appropriate network management. This includes tasks like incorporating new end devices into an existing network and combining them with other end devices to form functional units. In addition, a modern network management also offers mechanisms for the analysis and diagnosis of systems that are already up and running. Network management was not foreseen in the original OSI model; rather it was put into a dedicated OSI network management framework. It is to be understood as existing in parallel to the OSI layers as it affects all of them.

Physical and Data Link Layers

All end devices must, e.g., have the same channel configuration. This involves certain parameters that determine transmission on the underlying medium. In this connection, the applicable bit rate is of particular interest. Network management also concerns the repeaters operating on layer 1 and bridges (implemented in layer 2), because they can also be used to link subnetworks with different channel configurations.

Network Layer
Every end device or group of end devices must be provided with a unique address. With the help of intelligent routers, it is possible to form subnetworks and with that reduce the network load. These devices too can be contacted and then configured by the commands made available through the network management (such as loading of new routing tables).

Transport and Session Layers
These layers are mainly responsible for the service quality that is offered by the underlying network. Corresponding configuration possibilities, which can be carried out with the network management, are therefore closely linked to the term "quality."

Presentation Layer
Common syntax is a basic prerequisite for interoperability. Network management must inform two communicating end devices of the syntax of the matching data types.

Application Layer
Here, network management is concerned with the carrying out of tasks that relate to applications. It should be possible to load applications and define their specific configurations, but also to load and modify tables describing the communication relationships of applications.

Inside fieldbus protocols, network management is traditionally not very highly developed. This stems from the fact that a fieldbus normally is not designed for the setup of large, complex networks. There are again exceptions, especially in building automation, which consequently need to provide more elaborated functions for the setup and maintenance of the network. In most cases, however, the flexibility and functionality of network management is adapted to the functionality and application area of the individual fieldbus. There are systems with comparatively simple (ASi, Interbus, P-Net, J1939) and rather complex management functions (PROFIBUS-FMS, WorldFIP, CANopen, LonWorks, EIB). The latter are typically more flexible in their application range but need more efforts for configuration and commissioning. In any case, network management functions are normally not explicitly present (in parallel to the protocol stack as suggested by the OSI model), but rather directly included in the protocol layers (mostly the application layer).

20.5.3 Network Topologies

One important property of a fieldbus is its topology. Developers of fieldbus systems have been very creative in the selection and definition of the best suited physical layout of the network. Again, this selection was typically influenced by the target application area as well as by the available interface technologies that are used to build the fieldbus. Figure 20.12 shows the most relevant topologies for wired automation networks. It should be noted that the physical layer of a fieldbus has to meet quite demanding requirements like robustness, immunity to electromagnetic disturbances, intrinsic safety for hazardous areas, or costs. The significance of the physical layer is underpinned by the fact that this area was the first that reached (notably undisputed) consensus in standardization.

The "star" topology was the typical wiring in automation before the introduction of the fieldbus. The PLC was the center, attached to the distributed I/O elements with dedicated lines. The obvious cabling overhead was one of the main reasons to develop serial bus systems. With the adoption of switched Ethernet also for automation purposes, the star topology returned. Today, the central element is the Ethernet switch, and all Ethernet nodes are connected by means of a structured cabling, i.e., a dedicated link to each network node. Another application of the star topology is for fieldbus systems using optical fibers as transmission medium. Here, the center is an active star coupler linking

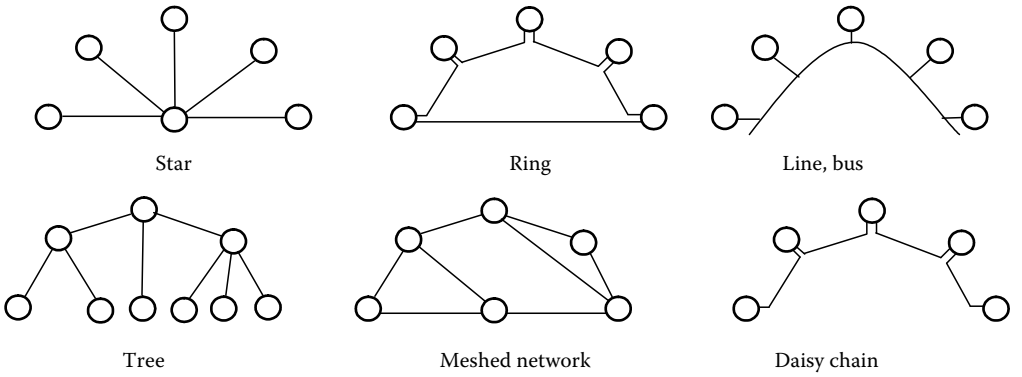


FIGURE 20.12 Topological network structures typically used in fieldbus systems.

the fibers (as in byteflight). Unless the coupler is a fully fledged node (i.e., addressable by the fieldbus protocol), this star topology is logically equivalent to a line structure.

Another simple topological form is the "ring." Here, each node has two network interfaces (an input and a separate, independent output), and the nodes are arranged one after another in the form of a chain. In its entirety the ring can be viewed as one large shift register, and it is usually also operated in this way (the most prominent example is INTERBUS). As there is no need to address the nodes explicitly, it is a very fast method to exchange data, and also a very deterministic one with low jitter (this is why SERCOS uses this topology for the interconnection of drives). A variant of the ring topology, actually a daisy-chain structure, was introduced with Industrial Ethernet. Here, each node contains a small switch, and the nodes are not connected in a star topology, but cascaded like a string of pearls. This layout is used, e.g., in PROFINET.

The "line," often referred to simply as "bus" (although this term is not unique), is the most successful and most commonly used network topology in the fieldbus world. It was the logical and most efficient replacement for the former starlike point-to-point cabling in that one single line should connect all network nodes (just as in the original concepts of Ethernet with the famous yellow cable and other types of coax cables). In many cases, the line topology is based on the RS 485 interface. This is an inexpensive, fully differential, multi-point interface standard using a shielded twisted pair cable with 120 Ω characteristic impedance. Maximum cable length is 1200 m, the maximum achievable data rate is 10 Mbit/s. The maximum number of nodes per segment depends on the electrical characteristics of the driver circuits and was originally limited to 32 (termed unit loads). Enhanced transceivers with higher input impedance meanwhile allow up to 256 nodes per segment. Beyond this, repeaters are necessary to regenerate the data signals. Some fieldbus systems (like CAN and all CAN-based systems) use RS 485 transceivers in a modified way (i.e., by essentially applying the data signal not to the actual data input but the output enable used to switch the output to a high-impedance state) to generate asymmetric bit patterns supporting a special form of medium access. A crucial aspect in practice is the proper electrical termination of the bus line to avoid signal reflections disturbing the data transfer. Wrong or missing termination is the most frequent problem leading to communication failures. A variant of the line topology used for P-NET therefore requires that the ends of the cable be connected, to form a closed loop. Nevertheless, electrically this structure is still a bus.

The "tree structure" is a composite network structure and characterized by one or more substations being dependent on a root node. Each substation can in turn be a root node for a lower-level segment. In many cases, the actual connections between the stations are regular point-to-point connections or lines. In automation technology, tree structures are the usual way to build hierarchical, relatively complex networks. The root nodes usually have routing capabilities, so that the data traffic can at least

partly be confined to individual areas of the network. This, however, requires caution in the planning phase of the network when defining communication relations between end nodes. Nodes exchanging lots of information should be kept in common subnets to avoid congestion on the backbone links. Tree structures are very common in building automation networks like EIB, LONworks, or BACnet.

Some fieldbus systems permit "free topologies," where nodes can be connected in a multi-point fashion without restrictions. Essentially, this is a variant of the line topology because the nodes are electrically interconnected. Contrary to the standard line topology, however, there are no stringent limitations concerning line length, the length of stubs, or branches that would cause severe signal degradation. This makes cabling very convenient for the installer, but poses a substantial challenge for the signal processing in the nodes and their transceivers. Fieldbus systems using such free topologies are, e.g., LONworks (where the free topology transceiver is the most widely used because of its robustness even in cases where conventional RS 485 transceivers would be sufficient), SERIPLEX, or ASI, which was designed for utmost simplicity and ease of handling in harsh industrial environments.

"Mesh" networks, where multiple paths through the network exist, play only a subordinate role in fieldbus systems because they require appropriate routing strategies to keep messages from circling in the network and causing congestion. LONworks and P-NET offer the possibility for building meshes.

The general topology used by the fieldbus also influences the way access to the medium is being handled. As a matter of fact, topologies were often selected according to the desired MAC method, or vice versa.

20.5.4 Medium Access Control

Fieldbus designers have been very creative in what concerns allocation of the available communication bandwidth to the individual nodes. On the data link layer, virtually all medium access principles also known from LANs are used, plus many different subtypes, refinements, and hybrid solutions (Figure 20.13). This has to be understood against the backdrop of desired efficiency. Resources both

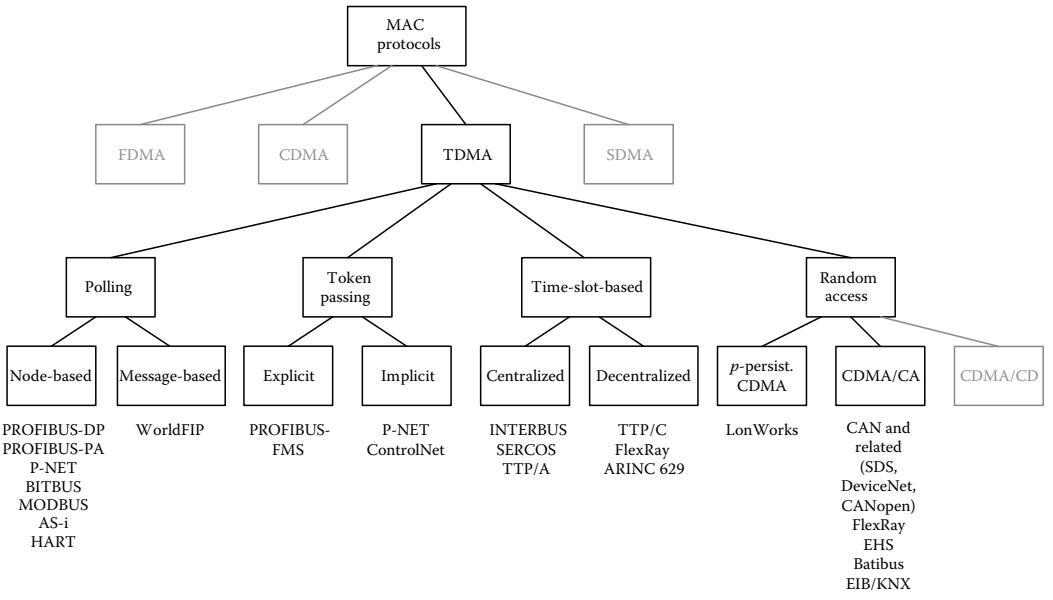


FIGURE 20.13 MAC strategies in fieldbus systems and examples.

in terms of computing power in the nodes and available data rate and thus bandwidth in the network were very limited at the time most fieldbus systems were invented. Hence, developers wanted to get the most out of what was at hand, and they found that a straightforward application of the medium access strategies used in computer networks was not efficient enough. So, they started designing new, more efficient methods tailored to individual application areas and optimized for the particular type of traffic and communication models they had in mind. In fact, the efficiency or overhead in the sense of how many bits must be transmitted over the network to convey a given amount of user data was a very common parameter to compare (and market) fieldbus systems—even though the significance of such a parameter alone was and is more than questionable.

An important classification for the way the data transfer is controlled is the distinction between single and multi-master systems. The single-master (or master–slave) approach reflects the tradition of centralized, PLC-based automation systems and is typically used for fieldbus systems in the lowest levels of the automation pyramid where the roles of the nodes in the network can be clearly distributed. Such networks typically have a limited size and a simple, mostly single-segment structure. The master either retrieves data from the slaves in direct request–response communication or explicitly synchronizes time slots that are then used by the slaves to send their data.

The alternative strategy is the multi-master approach, where all nodes are basically equal and must share the communication medium in a democratic and fair way. Such networks can have more complex structures, including hierarchical ones with large numbers of nodes, and are found mostly on the middle level of the automation pyramid (the cell level) and in building automation. Historically, multi-master fieldbus systems originated from the wish to have truly distributed systems for process control, and in many cases such fieldbus systems were more than pure communication networks—they were embedded in comprehensive operating-system-like software frameworks.

As far as the actual MAC strategies are concerned, all fieldbus systems use a time division multiple access (TDMA) strategy in the sense that the bandwidth is shared in the time domain, i.e., the network nodes use the communication line sequentially. Other multiplexing methods known in particular from telecommunications, such as frequency division multiple access, code division multiple access, or space division multiple access play no or only subordinate roles in the fieldbus world. More precisely, medium access is managed either in a centralized fashion by polling or time-slot-based techniques, or it is done in a decentralized way by token passing or random access methods. Within these basic methods, which will be briefly reviewed below, various variants and blends are in use.

20.5.4.1 Polling

Polling is a master–slave access scheme and foresees that a slave node is only allowed to send data when explicitly told so by a central master (sometime also called arbiter or arbitrator). On the network, there is a constant alternation between poll messages from the master to each slave and response messages from the slaves back. In its purest form, polling is strictly cyclic, which means that the master polls the entire list of slaves one by one and then restarts the cycle. The polling rate might be adapted (in theory even dynamically) if individual nodes have more data to transmit than others [142], but this is rarely done in practice.

Evidently, polling is perfectly suited for periodic traffic where all process variables need equidistant sampling (Figure 20.14). As the traffic is strictly controlled by one node, the cycle time can be kept constant provided that slaves implement anti-jabber mechanisms so that they cannot block the medium in the case of failure. Likewise, jitter can be kept low. Simple polling is employed in typical sensor–actuator bus systems like AS-I, HART, MODBUS, or lower-level fieldbus systems like BITBUS, PROFIBUS-DP, PROFIBUS-PA. Polling can also be used as an underlying bus access mechanism in a multi-master system, such that the permission to access the slaves is rotated between several master nodes. This strategy is used, e.g., by PROFIBUS-FMS or P-NET.

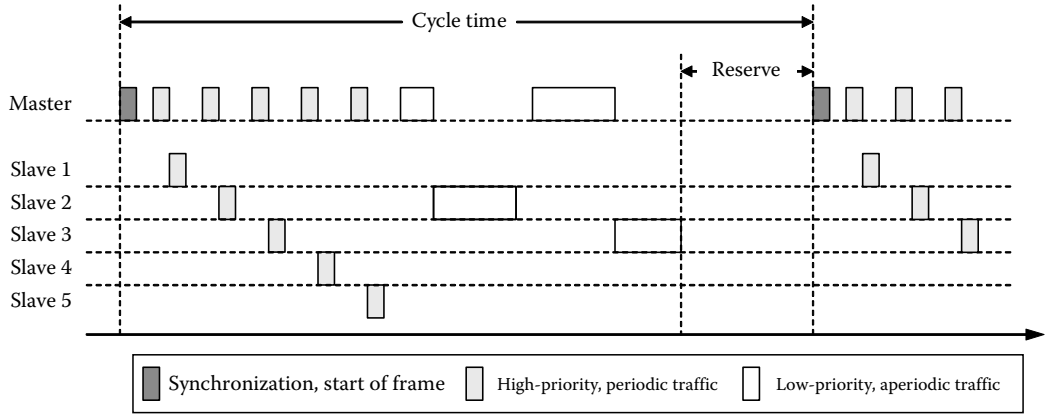


FIGURE 20.14 Bus cycle with cyclic polling and room for aperiodic traffic (PROFIBUS-DP V2).

A distinction between different flavors of polling can be made depending on the way data are addressed. The classical master–slave polling in the examples listed above uses explicit node addressing, i.e., the communication approach is device-centric. Another possibility is to identify the desired data by name, meaning that the request from the master concerns not a device, but a certain process variable. This is an approach employed by the producer/consumer model in WorldFIP, where the Bus Arbitrator polls a variable and the node generating ("producing") it then sends it onto the network, or in LIN, where the master retrieves data from the slaves by polling the message identifiers. The link active scheduler in Foundation Fieldbus performs a similar function by prompting devices to send their scheduled data to the consumers. This polling variant is sometimes called central polling. From the network traffic point of view, the two approaches are equal, each consists of a request–response pair of messages per variable exchange.

A peculiarity of the bus arbitration in WorldFIP, however, is that the polling mechanism accounts for different periodicity requirements of the individual variables. To this end, the polling cycle (here called macrocycle) is compiled at configuration time from several elementary cycles or microcycles in a static schedule to ensure that each variable will be sampled as often as needed under the constraint that the total data rate originating from this scheduled traffic is at any given time lower than the maximum data rate of the bus. The number of elementary cycles a macrocycle must have is essentially defined by the least common multiple of the periodicities divided by the highest common denominator of the periodicities (Figure 20.15).

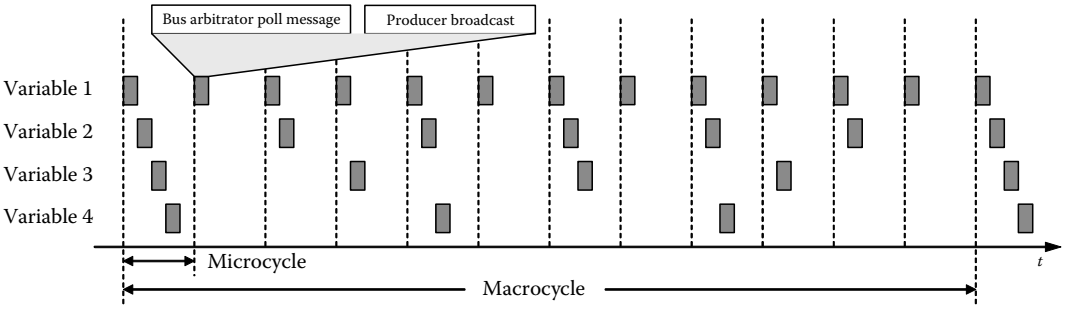


FIGURE 20.15 Bus cycle in WorldFIP.

In strict polling, there is no room for aperiodic traffic, which is a severe drawback for fieldbus systems designed to be more than simple sensor–actuator busses. Therefore, most fieldbus systems foresee some possibility for the master to handle sporadic data exchange for configuration data or process variables that require transfer only on an event basis. The important constraint is that this type of traffic must not interfere with the periodic one. In practice, this is done by reserving some bandwidth in addition to the scheduled periodic messages for a priori unknown aperiodic data transfer. PROFIBUS-DP/PA and many Ethernet-based automation networks (such as PROFINET) use a dedicated portion of the bus cycle after the periodic traffic for aperiodic traffic. To keep the cycle period constant, there is usually some spare time after this window to account for possible retransmissions of messages. In WorldFIP, aperiodic traffic can be scheduled in the empty slots of the elementary cycles, so that the macrocycle needs not be extended.

Another disadvantage of polling is that slaves cannot become active by themselves. If all of a sudden a slave—for instance in the event of an alarm condition—needs to transmit additional data, it first has to signal this alarm condition to the master to initiate an aperiodic request. In most cases (like in WorldFIP), the slaves have the possibility to set a flag in the ordinary cyclic data exchange messages to indicate the need for additional service (in the old CAMAC system, this flag had the descriptive name "look at me"). The master then typically starts a handshake procedure to retrieve these data, possibly by first requesting information about their amount to allocate an appropriate time slot for the transfer (also with respect to other pending requests). This basic mechanism can be further extended to allow a communication among slaves. Again, two possibilities exist. The first is to handle it indirectly via the master which gets the message from the sending slave and forwards it to the addressee (as done in the Measurement bus). This permits the master to maintain full control over the bus. The alternative is that the master delegates the bus access to the slave for a while until the slave has sent all its data and returns the access right to the master (as in WorldFIP, Foundation Fieldbus, or PROFIBUS-DP-V2). In this case, the master has to implement some timeout mechanism to ensure that the procedure is properly terminated.

20.5.4.2 Token Passing

One way to coordinate medium access between several peer stations in a network is token passing. In this method, the right to control the network is represented by a special piece of information called token, which is passed on from node to node. Only a node possessing this token may initiate data transfer. A set of rules ensures that the token is passed in a fair manner and that errors such as lost tokens or duplicate tokens are detected and resolved. Compared to time-slot mechanisms, token passing has the advantage that if a node has no data to send, it will hand over the token to the next station immediately, which saves time. Essentially, there are two ways of implementing the token: either explicitly by means of a dedicated short message or implicitly by distributed, synchronized access counters (AC) in all nodes. Token passing is often combined with an underlying master–slave mechanism to control a subset of nodes.

The explicit form of token passing is employed by PROFIBUS, both FMS and DP/PA (Figure 20.16). Even though PROFIBUS-DP in its basic variant is a single-master system, several DP networks may coexist on the same bus. A crucial point here is the correct setting of a timing parameter called target token rotation time T_{TR}. This parameter indirectly determines how long a master may occupy the bus. When a master receives the token, it starts a timer to measure the real token rotation time. When it receives the token the next time, it is allowed to exchange messages with other masters and slaves only while the timer has not reached the T_{TR}. As soon as this happens, the token must be handed over to the next master in the list. If a master receives the token after the timer is expired, it may send just one high-priority message before having to pass on the token again. The time available for each master therefore also depends on the amount of time all the other masters in the network hold the token, and it might vary from round to round if aperiodic data have to be sent or retransmissions

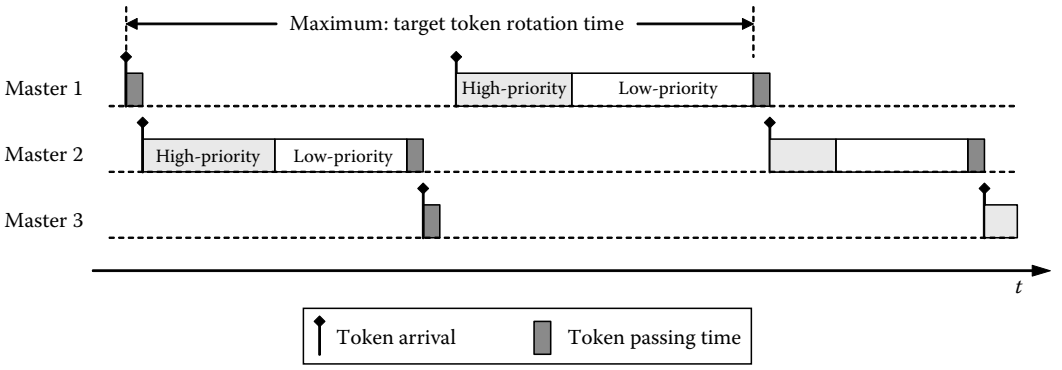


FIGURE 20.16 Token passing in PROFIBUS.

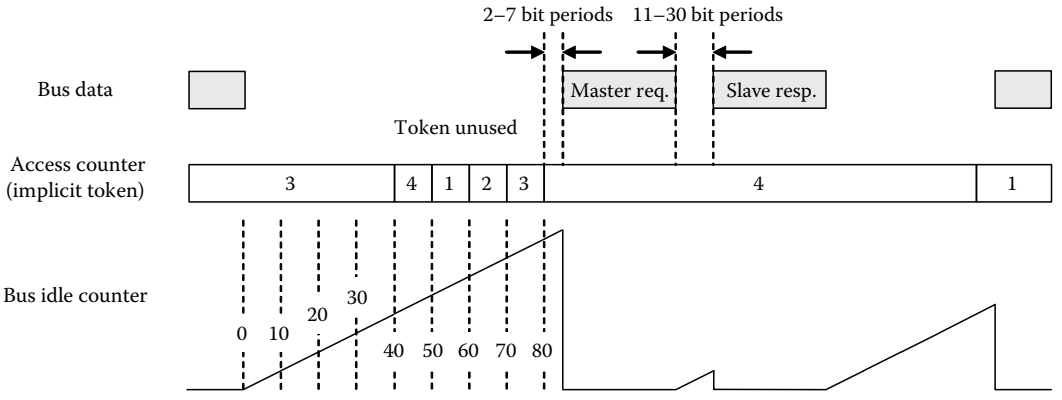


FIGURE 20.17 P-NET token passing and polling mechanism.

of individual frames are necessary. Periodicity of individual process variables is therefore less exact as with polling, and jitter may be larger. Still, the total time of a token rotation cycle can be upper-bounded to support real-time requirements. In any case, careful selection of the T_{TR} value must be done based on an a priori analysis of the expected network traffic and is essential to ensure optimal use of the available communication bandwidth.

The implicit form of token passing is used by P-NET. This fieldbus uses a hybrid approach together with polling. To keep the overhead of the volume of information that needs to be transferred on the bus to the absolute minimum, the token is simulated by two counters, included in every master, and is not actually passed around the bus (Figure 20.17). The "idle bus bit period counter" (IC) is incremented every bit period as long as the bus is free, and is reset as soon as some node sends data. The access counter (AC) is incremented whenever the IC reaches the values 40, 50, 60, . . ., 360. If the AC status matches the address of a master, this means that the master has the token and is therefore able to access the bus for one single request–response data exchange with some other node. To ensure that the token is not passed on, there are maximum values for the time a master can wait before sending a request to a slave, and also for the time a slave may take until it starts its response. If the master has no data to send, it remains silent, and after a further 10 increments, the token is handed over. Once the AC reaches the number of masters contained within the bus system, up to a maximum of 32, it is reset to the value 1. To avoid loss of synchronization due to the freely running system clocks in the nodes during long idle periods, a master has to send at least a synchronization

message if the IC value reaches 360. As the data exchange per master is strictly limited, the token rotation time is upper-bounded, even though it is likely not constant.

A similar implicit method is employed by ControlNet. A node possessing the token may transmit one single data frame possibly containing several link layer packets. All nodes monitor the source address of this frame, and at the end of the frame increment this address in an internal implicit token register. If the value of this register matches the address of a node, it possesses the token and may send its own data. If there are no data to be sent, the node must send an empty frame so that the token can be passed on. This mechanism is used both for scheduled and unscheduled traffic with two different tokens. While every node with scheduled traffic will have a guaranteed bus access every network cycle, the access right for unscheduled traffic changes in a round-robin fashion, i.e., the first node to get bus access for unscheduled traffic is rotated every cycle.

20.5.4.3 Time-Slot-Based Access

In time-slot-based methods, the available transmission time on the medium is divided into distinct slots, which are assigned to the individual nodes. During its time slot, a node may then access the medium. The difference to cyclic polling strategies—which in fact also come down to partitioning the polling cycle into time windows—is that the nodes are not requested by a central station to send their data; they can do it by themselves. Therefore, time-slot-based methods are mostly also called TDMA (in a narrower sense). The slots need not have the same size, they may be different. Likewise, they may be dynamically distributed to the nodes according to the amount of data they wish to send. This way, aperiodic traffic can be incorporated, whereas TDMA in general favors periodic traffic. A strictly equal distribution of the slots is also called synchronous TDMA and is particularly known from telecommunications technology, where each end device is assigned a time window in which, e.g., the correspondingly digitized voice data can be transmitted. There, such systems are referred to as "synchronous bus systems." The case where time slots are dynamically distributed is called asynchronous TDMA. Contrary to the synchronous version where the position of the time slot within the cycle implicitly identifies the sending nodes, some address information must be sent in the case of asynchronous TDMA.

The fact that nodes may access the medium without prior explicit request does not mean that TDMA methods are generally multi-master or peer-to-peer techniques. This depends on the way the synchronization of the time slots is achieved. Evidently, if no handshake procedures are implemented for data transfer, all nodes must be properly synchronized so that only one node at a time accesses the bus. This can be done in a centralized or decentralized way.

The simpler version is to have one dedicated bus master sending some sort of synchronization message at the start of the cycle. After that, the nodes exchange their data in their previously assigned time slots. Such a strategy is used, e.g., by SERCOS, which is therefore regarded as a master–slave fieldbus (Figure 20.18). In this particular case, the time slots for the slaves are all equal in length, and the master has a larger time window to send its data to the slaves afterwards. A similar method is also used by TTP/A. Here, a bus master (apart from having the possibility for simple polling) sends a synchronization message in the first time slot of a round to start the cyclic data transfer from all other nodes. In byteflight, a node (the sync master) sends a periodic synchronization signal to start the data exchange round (in byteflight called slot), and the others initialize slot counters to determine the position of their predefined microslot. If a node has no data to send, the respective microslot is shortened. In this respect, the bus access mechanism resembles an implicit token passing method. The spare time of unused microslots becomes available for aperiodic or lower-priority traffic wards the end of the round.

Related to the centralized TDMA approaches is the summation frame protocol of INTERBUS (Figure 20.19). As noted in the previous section, the system is based on a ring structure and acts like a large shift register. Each slave has a predefined slot in the data frame that is selected in

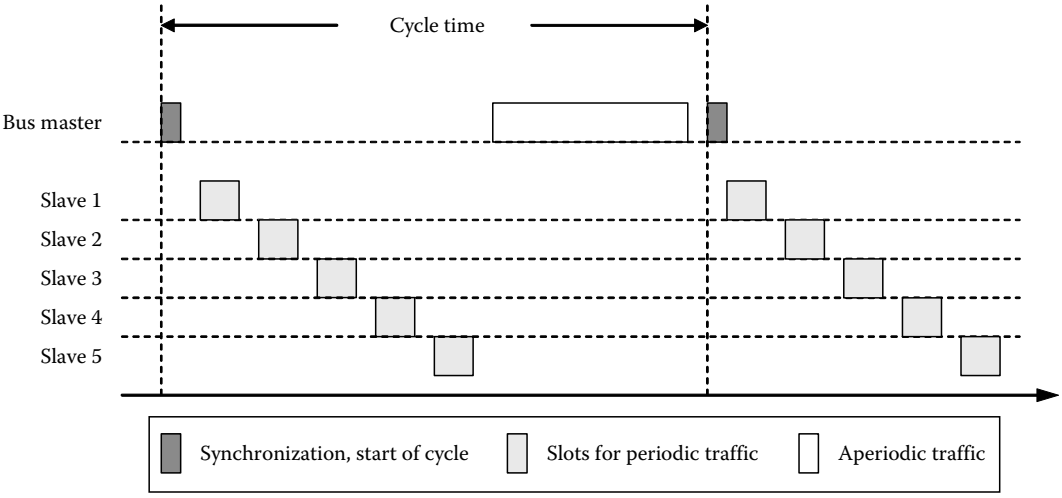


FIGURE 20.18 Centralized TDMA scheme (e.g., SERCOS).

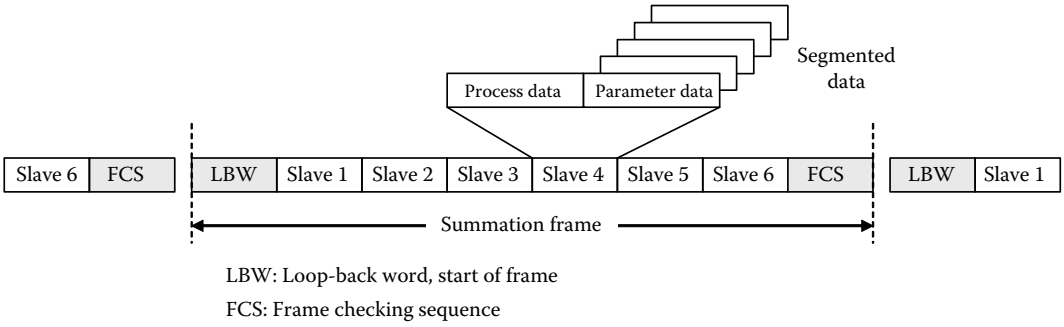


FIGURE 20.19 Summation frame of INTERBUS.

accordance with its position in the ring. Output data from the master to the slave and input data from the slave back are exchanged in the respective slot as the frame is shifted through the ring. Thus one data frame is sufficient for the cyclic updating of all end devices. A clever arrangement of buffers furthermore ensures that despite the ring structure with its inherent delays, all I/Os at the slaves are updated at the same time, so that synchronous operation with respect to the process variables is guaranteed. Aperiodic traffic can be introduced in this rather rigid scheme through a so-called parameter channel. Larger amounts of data are transferred in small packets in this channel without affecting the exchange of cyclic process data.

The second method of managing the time slots is a decentralized one. In this case, there is no dedicated node to initiate the cycle, rather all devices synchronize themselves either by explicit clock synchronization mechanisms or by a set of timers that settle bus operation down to a stable steady state. Fieldbus systems relying on such distributed mechanisms have a high degree of fault tolerance because there is no single point of failure as in centralized approaches. Therefore, they are well suited for safety-critical applications (such as TTP/C, FlexRay, or ARINC 629), provided that they are designed to have real-time capabilities. Nevertheless, the underlying algorithms are relatively complex, which is a why such systems have attracted a lot of scientific interest. The distributed nature

of bus access requires proper error containment mechanisms to avoid faulty nodes (babbling idiots) from blocking the medium and jeopardizing real-time behavior.

The real-time properties of time-slot mechanisms also led to the introduction of such approaches in fieldbus systems that originally use different access control methods. As prominent example, CAN was enhanced in this way by superimposing TDMA structures like in time-triggered CAN (TT-CAN) or flexible time-triggered CAN (FTT-CAN). Last not least, it should be noted that the boundaries between TDMA and other MAC techniques are sometimes blurred. For instance, the access control method of ControlNet is called concurrent time domain multiple access, although it is rather an implicit token passing strategy. The flexible time domain multiple access (FTDMA) of byteflight also has some similarity to token passing.

20.5.4.4 Random Access

Random access basically means that a network node tries to access the communication medium whenever it wants to—without limitations imposed for instance by any precomputed access schedule. This principle is called carrier sense multiple access (CSMA), it was first implemented in the ALOHA network in 1970 and since then has been modified in various ways. Obviously, it is perfectly suited for spontaneous, peer-to-peer communication. Its conceptual simplicity comes at the expense of a severe drawback: collisions inevitably occur when several nodes try to access the network at the same time, even if they first check if the line is idle. The variants of CSMA therefore have one common goal—to deal with these collisions in an effective way without wasting too much communication bandwidth and thereby avoiding excessive communication delays.

The best-known CSMA variant is CSMA-CD (Collision Detection) used in Ethernet, more precisely in its original version with shared medium. Here, collisions are immediately detected by the sending nodes which monitor the bus while sending. After CD, the nodes abort the data transfer and wait for a random time before trying again. In fieldbus systems, this variant is not very common because despite its stochastic component, there is a high probability that collisions remain if the source data rates are too high. In practice, the effectively attainable throughput is limited to values well below 50% of the maximum data rate, which was always deemed insufficient for automation networks. It should be noted, though, that things changed with the introduction of switched Ethernet.

A variant where the backoff time (the time a node waits before a retry) is not just random, but adaptable is called predictive p-persistent CSMA and used in LonWorks. Here, p denotes the probability that the node will try again in a certain time interval after the collision. This probability can be adapted by each node based on an estimation of its backlog and the monitored network load. In high-load conditions, the nodes reduce their probability for starting a medium access, which in turn reduces the probability for collisions. LonWorks additionally has a number of high-priority "time slots" with differing probabilities for medium access, so that messages with a higher priority still have a better chance to be sent without long delays.

The most widely used CSMA strategy in the fieldbus world, however, uses asymmetric symbols for coding the bits on the bus line, so that when two different bits are sent at a time, one of them wins over the other. In this case, there is no actual collision, bus access is nondestructive, and this is why this strategy is called CSMA-CA (Collision Avoidance), sometimes also called CSMA-BA (Bitwise Arbitration). The first fieldbus to use this method was CAN, and this example will be used in the sequel to explain the idea in more detail.

The bus line is designed as an open collector bus so that the low level is "dominant" and the high level remains "recessive." This means that a "1" sent from an end device can be overwritten by a "0." CAN uses message-based addressing, and after sending a synchronization bit, the nodes write the identifier of their message to the bus bit by bit and at the same time observe the current bus level. If the bits sent and read back differ, arbitration is lost, and the node stops sending. In the end, the

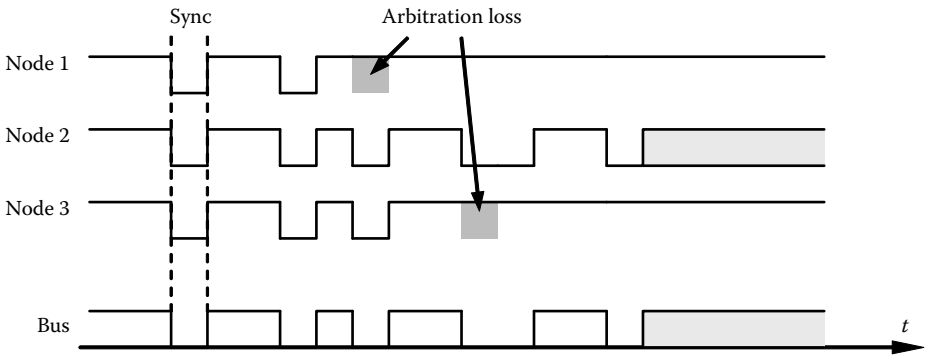


FIGURE 20.20 BA method with CAN.

remaining node is the one whose message has the lowest identifier (Figure 20.20). The BA method brings with it a major disadvantage. The propagation time of the signals on the line must be short compared to the bit time to yield quasi-simultaneity for all nodes. With the highest bit rate of 1 Mbit/s, this means a maximum bus length of only 40 m. In motor vehicles (the original application field of CAN), this restriction is of lesser importance but in industrial automation technology it can lead to a reduction in the data transfer rate.

As with decentralized TDMA, faulty nodes may infinitely block the bus. To prevent this situation, CAN nodes contain error counters which are incremented whenever the node detects transmit or receive errors and are decremented after successful transmit or receive procedures. The counter status reflects the reliability of the node and determines if the node may fully participate in bus traffic or only with certain restrictions. In the extreme case, it is completely excluded from the bus.

BA was so successful that it was used in several other fieldbus systems in similar form. Examples are building automation networks like EIB, BATIBUS, or EHS, as well as in other automotive networks like VAN and FlexRay (for aperiodic traffic). But also CAN was used as a basis for further extension. CAN as such originally only defined layers 1 and 2 in the OSI model. Although this was sufficient for the exchange of short messages within a closed network, it was insufficient for automation technology. For this reason, the CAN-in-automation user group (CiA) defined the CAN application layer (CAL) and then the CANopen protocol. Other protocols for automation technology, also based on CAN, are DeviceNet and SDS. The CAN Kingdom protocol has been specially developed for machine controls and safety-critical applications. These higher-level protocols offer the possibility of exchanging larger volumes of data and of synchronizing end devices. Network management functions solve the problems of node configuration and identifier specification.

CSMA-CA has one inherent problem. Even in the absence of babbling idiots, the highest priority object can practically block the bus and messages with lower priorities seldom get through. Therefore, QoS guarantees can only be given in a strict sense for the message with the highest priority if no additional measures are taken. A number of solutions have been proposed to overcome this problem. One of them is to limit the frequency of the messages such that after a successful transmission, a node has to wait a certain amount of time before being able to send again. Another possibility is to introduce mechanisms in the upper protocol layers to exchange identifiers cyclically within specific priority classes in addition to restrictions in access frequencies. This is a network management function used, e.g., in CANopen. The third method of improving fairness (and thus real-time capabilities) is to superimpose time-slot mechanisms at least for some message and traffic classes. This strategy is being employed by TT-CAN and FTT-CAN.

20.5.5 Communication Paradigms

It was stated before that there are many diverse types of traffic to be considered in fieldbus systems, and there are many different services that a fieldbus should provide to the user application. Apart from basic functions like reading and writing variables, management of objects is required. This includes updates of device configurations, starting and stopping tasks running on nodes, up- or downloading of programs, triggering and synchronization of events, establishment and termination of connections between nodes, access control to node data, to name just of few. In general terms, fieldbus services need to handle objects by identifying them, the actions to be performed on them, as well as the appropriate parameters for the actions. These services must be defined and provided by the upper layers of the protocol stack, in most cases by the application layer.

Communication and cooperation between different application layers and applications in a fieldbus system are based on a basic set of two or three different models [139]. These models again represent different philosophies of the type of information that is exchanged between two or more entities. One approach is to build the cooperation on actions or functions into which a more complex process is decomposed. The responsibility for interpretation of the information is largely concentrated on the sender's side. This is the philosophy behind the client–server model. The other approach is a data-oriented one. Here, not actions but data are exchanged, and the responsibility for their interpretation is with the receiver (which then might take according actions). This is the idea behind the publisher–subscriber model and the producer–consumer model. The most relevant properties of these three are summed up in Table 20.1.

The overview shows that communication paradigms need to be supported by the medium access mechanism of the data link layer to achieve optimal performance.

20.5.5.1 Client–Server Model

In the client–server paradigm, two entities cooperate in the information exchange. The entity providing a service or data is called server, the one requesting the service is called client. In general, the server may become active only upon request from the client, which means that this model is better suited for state-based data traffic handled in some scheduled manner. Events, i.e., spontaneous traffic, can be handled only if they occur at the client (which may solely initiate communication). If they occur at the server, they cannot be handled in a spontaneous way at all; the server has to postpone the transmission until it receives an appropriate request from the client.

Client–server communication is based on confirmed services with appropriate service primitives (request, indication, response, confirm) as defined in the OSI model (Figure 20.21, left side). This is the classical implementation used by many fieldbus systems which derive their application layer protocols essentially from the MMS standard. Examples are PROFIBUS-FMS (DP and PA in reduced form as well), WorldFIP, INTERBUS, or P-NET. The MMS model proposed comprehensive services that are usually only partly implemented in the fieldbus application layers. For the example

TABLE 20.1 Properties of Communication Paradigms

	Client–Server Model	Producer–Consumer Model	Publisher–Subscriber Model
Communication relation	Peer-to-peer	Broadcast	Multicast
Communication type	Connection oriented	Connectionless	Connectionless
Master–slave relation	Monomaster, multi-master	Multi-master	Multi-master
Communication service	Confirmed, unconfirmed, acknowledged	Unconfirmed, acknowledged	Unconfirmed, acknowledged
Application classes	Parameter transfer, cyclic communication	Event notification, alarms, error, synchronization	State changes, event-oriented signal sources (e.g., switches)

Source: Sauter, T., in *The Industrial Communication Handbook*, CRC Press, Boca Raton, FL, 2005, 7.1–7.39. (With permission.)

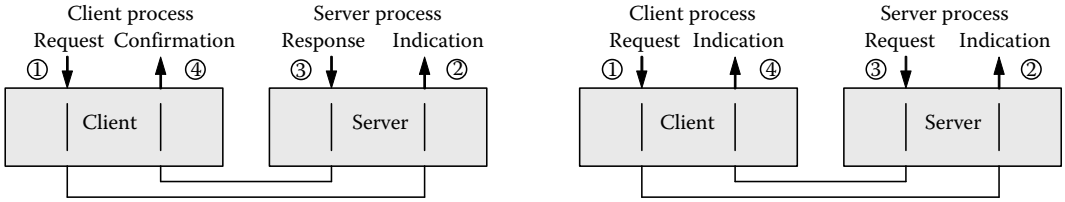


FIGURE 20.21 Communication services used in the client–server model (left: normal; right: based on unconfirmed services).

of PROFIBUS-FMS (in similar form they exist also in Foundation Fieldbus and WorldFIP), the services comprise the following groups: variable access (read, write), transmission of events, execution of programs, domain management (up- or download of large amounts of data), context management (administration of communication relationships), and object dictionary management (administration of the list of all objects in a device). Depending on the actual service requested, the semantics of the information exchanged obviously differ.

In some cases, and deviating from the standard model described before, a client–server communication can also be built from two unconfirmed services (Figure 20.21, right side). This means that the response generated by the server is not related to the request by means of the underlying protocols (i.e., it does not generate a confirmation on the local application layer). Rather, the client application has to keep track of the request and needs to relate the indication generated by the response on its own. This mechanism reduces the implementation efforts on the protocol side at the expense of the application complexity. Another—nonstandard—variant of the client–server model may have several responses from the server answering a single request [139]. Such an approach can be beneficial if the service execution on the server side takes a long time, so that partial results are reasonable or at least indications about the status of the execution. This may help circumvent problems with, e.g., hard-coded timeouts in legacy client applications by means of some keep-alive signal. Somewhat related to this and resembling a publisher–subscriber-type operation is a multi-response from a server following one single request from the client. This can be useful if strictly periodic data transmissions from the server are required. Contrary to a cyclic client–server polling strategy which always is affected by network and software delay, the server can control the sampling and transmission period of the data to be sent in a better and more accurate way.

As far as the interrelation of the client–server model and MAC strategies is concerned, it should be noted that basically, a client–server-type communication can be implemented in both mono- and multi-master systems. In the latter cases (which may be based on CSMA, TDMA, or token passing), every master can take on the role of a client, whereas in mono-master systems (polling-based or centralized TDMA) this position is reserved for the bus master. Consequently, the client–server paradigm is used mainly for master–slave systems (as represented by PROFIBUS, AS-I, MODBUS, P-NET, BITbus, INTERBUS) and for reliable data transfer on application level (e.g., for parameterization data, file transfer, network and device management). In particular for management functions, the client–server model is widely used also within fieldbus systems that organize their regular data transfer according to the publisher–subscriber model, such as WorldFIP, EIB, CANopen, DeviceNet, ControlNet, or LonWorks.

20.5.5.2 Publisher–Subscriber Model

The basic idea of the publisher–subscriber model is that certain nodes (the publishers) produce information which they post into the network. The subscribers are typically groups of nodes that listen to information sources. Relating publishers and subscribers can be done at runtime. The

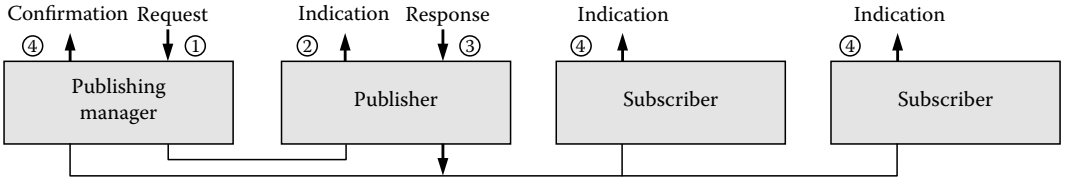


FIGURE 20.22 Pull-type publisher–subscriber model.

producer–consumer model uses very similar mechanisms, the only difference is that broadcast services are being employed instead of multicast communication as in the case of the publisher–subscriber model. However, this distinction is rather subtle and not very relevant in practice.

Depending on how the information exchange is initiated, two different subtypes of the publisher–subscriber paradigm can be distinguished [69]. In the pull model, the publishing action is triggered by a centralized publishing manager (Figure 20.22). Upon receiving the request from the manager, the publisher broadcasts the respective response to the network. Filtering of the message, which determines if the service is a broadcast or multicast, is done locally by the subscribers who listen into the network for message identifiers they subscribe to. It should be noted that the subscriber group may as well consist of just one node, such that the communication relation is effectively a 1-to-1 relationship.

In terms of communication services, the pull publisher–subscriber model uses a confirmed request/response service for the interaction between manager and publisher. The peculiarity of this service, however, is that while the request is unicast (i.e., has only one recipient), the response from the publisher is multicast to the publishing manager and the subscribers and already contains the information to be published. It is the task of the underlying protocol layers to ensure that the appropriate addressing scheme is used to transform the unicast into a multicast message. For the subscribers, the communication then may look like an unconfirmed service (i.e., they receive only an indication that a subscribed object has been transmitted) although it is actually a confirmed one from the viewpoint of the manager.

Still, there might be subtle differences in the way the request itself is handled. On the one hand, the manager might know the identifier of a variable and requests publication of this specific variable without caring about its producer. This method is being used, e.g., in WorldFIP. On the other hand, the manager might know the device address of the publisher and send the request directly to this node. This is typically done in fieldbus systems that normally rely on node addressing for data exchange and thus a client–server model. In such fieldbus systems, a publisher–subscriber transaction is the exception rather than the rule, as in PROFIBUS-DP V2. In master–slave systems, this type of communication—although not strictly publisher–subscriber—may also be used to implement a direct communication possibility between slaves under the direct control of the master. This shows that an application-layer service for reading data based on the pull model comes very close to a read service implemented according to the client–server model. The essential difference is that the response is directed to a group of receivers instead of only one—the manager.

The second subtype of the publisher–subscriber paradigm is the push model (Figure 20.23) in which the publishers become active by themselves without prior request from a centralized station and distribute their information to the subscribers when they consider it necessary, e.g., triggered by the expiration of a timer (as in TDMA systems) or by an external event. In terms of communication services, the publishing of information is implemented by means of unconfirmed (or locally confirmed) services on the publisher's side. Like in the pull variant, the subscribers receive an indication that new data have arrived, and do not answer or acknowledge receipt of the message.

A necessary step in either of the two variants is the subscription of the subscribers with the publisher. This is a typical configuration action within the network management that can be accomplished

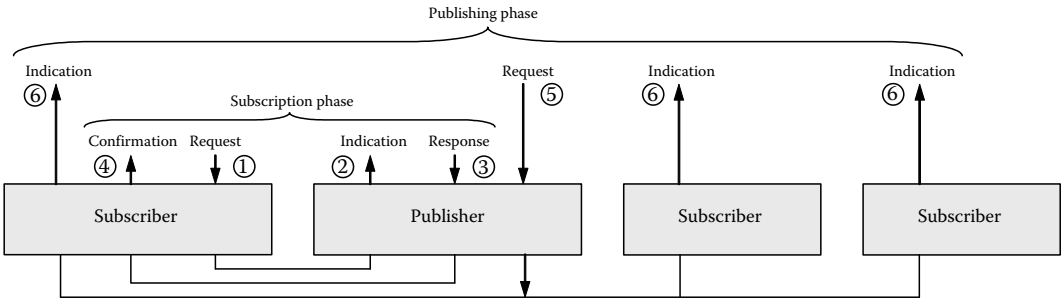


FIGURE 20.23 Push-type publisher–subscriber model.

statically during system setup and commissioning or dynamically during runtime. As the underlying communication mechanism of the publisher–subscriber model is a multicast to a defined group of nodes, it must be ensured that a node subscribing to a given message or object joins the correct communication group. This is therefore mainly done using client–server-type communication based on confirmed services.

Processes with mostly event-based communication can get along very well with publisher–subscriber- or producer–consumer-type communication systems. Depending on the fundamental layer-2 communication methods, real-time requirements may be more (in the case of TDMA methods or centralized polling) or less met (with pure random access methods). The obvious advantage is that all connected devices have direct access to the entire set of information as the broadcasting is based on identification of messages rather than nodes. Reaction times on events can be very short due to the absence of slow polling or token cycles. Generally, publisher–subscriber-type systems (or subsystems) are mostly multi-master systems because every information source (producer) must have the possibility to access the bus. The selection of relevant communication relationships is solely based on message filtering at the subscriber's side. Such filter tables are typically defined during the planning phase of an installation. It is no wonder that this communication model is being widely used in fieldbus systems, as for instance WorldFIP, Foundation Fieldbus, CAN, CANopen, DeviceNet, ControlNet, LIN, EIB, or LonWorks.

Still, there is an inherent problem. As both paradigms are message-based and therefore connectionless on the application layer, they are not suited for the transmission of sensitive, non-repetitive data such as parameter and configuration values or commands. Connectionless mechanisms can inform the respective nodes about communication errors on layer 2, but not about errors on application layer. Particularly for fieldbus systems devised for safety-critical applications (such as TTP), additional mechanisms have been developed to enable atomic broad- and multicasts. Atomic in this context means that the communication is successful only if all subscribers receive the distributed information correctly. Only in this case will actions associated with the transaction be executed, otherwise they are canceled.

20.5.6 Above the OSI Layers—Interoperability and Profiles

A key point for the acceptance of fieldbus systems was their openness. The availability of publicly available specifications and finally standards made it possible to set up so-called multi-vendor systems where components developed by different manufacturers work together to achieve a common functionality. This possibility was the evidence for interoperability and still is an important argument in fieldbus marketing. The standardization of fieldbusses was originally thought to be sufficient for interoperable systems, but reality quickly showed that it was not. Standards (interoperability

Interoperability level / Necessary agreements	Incompatible	Compatible	Interconnectable	Interworkable	Interoperable	Interchangeable
Temporal behavior						
Functional behavior						
Data semantics						
Definition of the data						
Access to the data						
Protocol (layers 1–7)						

FIGURE 20.24 Application functionalities within the interoperability definitions.

standards, to be precise) define the rules that the system components must comply with to be able to work with other components, but they often leave room for interpretation, and actual implementations may vary. Certification of the devices is a suitable way to reduce the problems, but by no means a guarantee. Another reason for troubles is that the semantics of data objects are not precisely defined. In fact, there are various degrees of interoperability, as defined by the IEC TC65 SC65C WG7 within the scope of the IEC "Interoperability Definitions" (Figure 20.24). These definitions relate to application functionalities that can be achieved, i.e., the degree to which devices and applications can actually work together.

Incompatible. The two communication partners have no common ground and their connection, if physically possible, could result in damage. The connectors, voltage levels, and modulation methods can be different.

Compatible. The most fundamental requirement is that both devices use the same communication protocol. They can be connected and may exchange data packets. Still, compliance with the protocol (as verified, e.g., by conformity tests according to ISO 9646) alone does not guarantee meaningful cooperation between the applications as they cannot make sense of the received packets.

Interconnectable. The partners have the same methods of accessing the data, and they mutually understand the addressing methods used by the partner. This is important because the exact way in which data are made available in the network is not defined by the protocol layers; they just offer a number of services for data transmission. Being interconnectable, the partners can recognize whether received messages were addressed to them, and they are capable of extracting the data (or encoding them in the opposite direction). Furthermore, they use the application layer services in the same way and have a common understanding of data types use for their applications.

Interworkable. Applications share the same definitions of data, and they use the same variable types. Therefore, they can correctly exchange data using a common way of coding. As an example, a sensor can send its value in long_int format, and a controller can correctly receive and process this value. Usually, the development tools offer support for this level. If the tools are designed for distributed systems, errors violating interworkability can already be detected in an early development phase.

Interoperable. On this level, the semantics of the data play an important role. Variables take on physical relevance; they do not just have a data type, but also a physical unit that needs to be known to the partners. Furthermore, accuracy ranges, meaning, and purpose of variables are defined. If a device produces a larger data set, it is known how the individual data can be distinguished. Interoperable nodes can be used immediately without additional description, their network interfaces are unique, and also the functional behaviors match in that is precisely defined what happens to the data that are sent across the network in the sense of which actions or sequence of actions they trigger.

Interchangeable. Even the temporal behavior of two nodes is the same with respect to the needs of the application. Typically, such requirements are maximum reaction times to events or commands. If such constraints are not equal, the overall behavior of the entire network can fundamentally change even if a device replacing another is functionally equivalent.

The problem of interoperability has been disregarded in many cases until recently. In fact, it is not a problem of the fieldbus itself, but of the application. Consequently, it must be tackled beyond the ISO/OSI model. The definition of appropriate "profiles" addresses this problem. A profile defines which variables carry which data, how they are coded, what physical units they have, etc. By virtue of this profile, it is possible to subject devices, which are claimed to satisfy this profile, to a corresponding conformity test. The compatibility of a device is generally also tested when used within a multi-vendor system.

The creation of profiles originated from the recognition that the definition of the protocol layers alone is not sufficient to allow for the implementation of interoperable products, because there are simply too many degrees of freedom. Therefore, profiles limit the top-level functionality and define specialized subsets for particular application areas [47]. Likewise, they specify communication objects, data types, and their encoding. So they can be seen as an additional layer on top of the ISO/OSI model, which is why they have also been called "Layer 8" or "User Layer." One thing to be kept in mind is that nodes using them literally form islands on a fieldbus, which contradicts the philosophy of an integrated, decentralized system. Different profiles may coexist on one fieldbus, but a communication between the device groups is normally very limited or impossible at all.

The concept of profiles has many names. In MMS, they are termed Companion Standards. In P-NET, they are equivalent to the Channel concept. Function blocks in Foundation Fieldbus essentially pursue the same idea. In LonWorks, there are the Standard Network Variable Types, and in EIB, interoperability is achieved by means of the EIB Interworking Standards.

From a systematic viewpoint, profiles can be distinguished into communication, device, and branch profiles. A bus-specific "communication profile" defines the mapping of communication objects onto the services offered by the fieldbus. A "branch profile" specifies common definitions within an application area concerning terms, data types, their coding, and physical meaning. "Device profiles" finally build on communication and branch profiles and describe functionality, interfaces, and in general the behavior of entire device classes such as electric drives, hydraulic valves, or simple sensors and actuators.

The work of defining profiles is scattered among different groups. Communication profiles are usually in the hands of fieldbus user groups. They can provide the in-depth know-how of the manufacturers, which is indispensable for bus-specific definitions. Device and branch profiles are increasingly a topic for independent user groups. For them, the fieldbus is just a means to an end—the efficient communication between devices. What counts more in this respect is the finding and modeling of uniform device structures and parameters for a specific application. This forms the basis for a mapping to a communication system that is generic within a given application context.

The ultimate goal is the definition of fieldbus-independent device profiles [47]. This is an attempt to overcome on a high level the still overwhelming variety of systems. Finally, such profiles are also

expected to facilitate the employment of fieldbus systems by the end-user who normally is only concerned about the overall functionality of a particular plant—and not about the question which fieldbus to use. The methods used to define data types, indices, default values, coding and meanings, identification data, and device behavior are based on functional abstractions (most promising are currently function blocks [43,48]) and general modeling techniques [49].

20.5.7 Fieldbus Management

Owing to the different capabilities and application areas of fieldbus systems, the management of a fieldbus shows varying complexity and its solutions are more or less convenient for the user. It has already been stated that the various fieldbusses offer a wide range of network management services with grossly varying levels of sophistication. Apart from the functional boundary conditions given by the protocols, fieldbus management always strongly relies on the tool support provided by the manufacturers. This significantly adds to inhomogeneity of the fieldbus world in that entirely different control concepts, user interfaces, and implementation platforms are being used. Furthermore, a strict division between communication and application aspects of fieldbus management is usually not drawn.

Typical "communication-related" management functions are network parameter settings like address information, data rate, or timing parameters. These functions are rather low level and implicitly part of all fieldbus protocols. The user can access them via software tools mostly supplied by the device vendor. "Application-related" management functions concern the definition of communication relations, system-wide timing parameters (such as cycle times), priorities, or synchronization. The mechanisms and services offered by the fieldbus systems to support these functions are very diverse and should be integrated in the management framework for the application itself (e.g., the control system using the fieldbus).

As a matter of fact, a common management approach is still not available despite all interoperability achievements, and vendor-specific solutions are preferred. From the user's point of view (which includes not only the end-users, but also system integrators), this entails significantly increased costs for the build-up and maintenance of know-how because they must become acquainted with an unmanageable variety of solutions and tools. This situation actually revives one of the big acceptance problems that fieldbus systems originally had among the community of users: the missing interoperability. Communication interoperability (as ensured by the fieldbus standards) is a necessary but not sufficient precondition. For the user, "handling interoperability" of devices from different vendors is equally important. What is needed are harmonized concepts for configuration and management tools. More than that, tools should also be consistent for different aspects of the life cycle of an installation, like planning, configuration, commissioning, test, and diagnosis or maintenance. Such tools require functionality-oriented, abstract views. A major disadvantage of today's tool variety is that they operate in many cases on incompatible data bases, which hampers system integration and is likely to produce consistency problems. More advanced concepts build on unified data sets that present consistent views to the individual tools with well-defined interfaces [94,95]. The data structures are nevertheless still specific for each fieldbus.

For a fieldbus-independent access to the field devices and their data (not necessarily covering the entire life cycle), several solutions have been proposed. They mostly rely on a sort of middleware abstraction layer using object-oriented models. Examples are OLE for process control (OPC) [96], also Java or other concepts [97]. Such platforms can ultimately be extended through definition of suitable application frameworks which permit the embedding of generic or proprietary software components in a unified environment spanning all phases of the life cycle. Relevant approaches are, e.g., Open Control [95], Field Device Tool [98], or a universal framework of the ISO [99].

In the context of management and operation frameworks, the unified description of device and system properties becomes of eminent importance. To this end, device description languages were

introduced. Over the years, several mutually incompatible languages and dialects were developed [101,102]. Originally, they were tailored to individual fieldbus systems (e.g., HART DDL, PROFIBUS GSD, CANopen EDS, FF DDL) and laid the foundation for user-friendly configuration and engineering tools. In recent years, the diversity of description languages is being addressed by the increased usage of universal languages like XML [103,104], which is also the basis for the electronic device description language (EDDL) standardized in IEC 61804 [105,106].

With the increasing importance of LAN and Internet technologies in automation, new approaches for fieldbus management appeared that may be apt to introduce at least a common view at various fieldbusses. All these concepts aim at integrating fieldbus management into existing management applications of the higher-level network, which is nowadays typically IP-based. One commonly employed high-level network management protocol is simple network management protocol (SNMP), which can be used to access also fieldbus data points [51,52]. Another approach involves the use of Directory Services [53]. These two solutions permit the inclusion of a large number of devices in specialized network management frameworks. An alternative that has become very popular is the use of Web technology, specifically HTTP tunneled over the fieldbus, to control device parameters. This trend is supported by the increasing availability of embedded Web servers and the use of XML as device description language [54]. The appealing feature of this solution is that no special tools are required and a standard Web browser is sufficient. However, Web pages are less suitable for the management of complete networks and rather limited to single-device management. Nevertheless, this approach is meanwhile pursued by many manufacturers.

20.5.8 Quest for Unification—The NOAH Approach

Standardization bodies recognized quite early that the heterogeneity of current fieldbus solutions was a problem. While the IEC standardization project still fought for a unified solution, CENELEC in Europe stimulated a joint research project supported by the European Commission under the ESPRIT programme. The goal of network-oriented application harmonization (NOAH) was to overcome the incompatible user interfaces of existing standardized fieldbus systems and to provide a uniform access layer on top of the individual technologies, thereby enabling true distributed automation systems in multi-vendor environments. The explicit intention of the project was the achievement of results that could then directly be pursued as standards. Originally, the focus of the project was the industrial fieldbusses contained in EN 50170 (P-Net, PROFIBUS, WorldFIP), later the work was extended to other systems as well (Foundation Fieldbus, HART, CAN). In fact, NOAH and its predecessor projects ACORN and RACKS were a comprehensive endeavor to tackle three basic aspects in networked automation systems [48]:

- Independent communication interfaces not only for operational services, but also for system and network management, i.e., for the setup and maintenance of an automation network.
- Device harmonization through the definition of device profiles that specify certain functions and parameters. The devices provide these functions and parameters for other profile devices or for control, visualization, supervision, maintenance, and technical management purposes. Within the scope of this aspect also a new, unified modeling language (UML)-based device description language (EDDL) was developed after an investigation of several description languages that existed at that time and were found to be too limited in capabilities.
- System integration in terms of a single and consistent data repository containing all device and network information [50] needed for a complete system life cycle including design, engineering, installation, configuration, maintenance, and management.

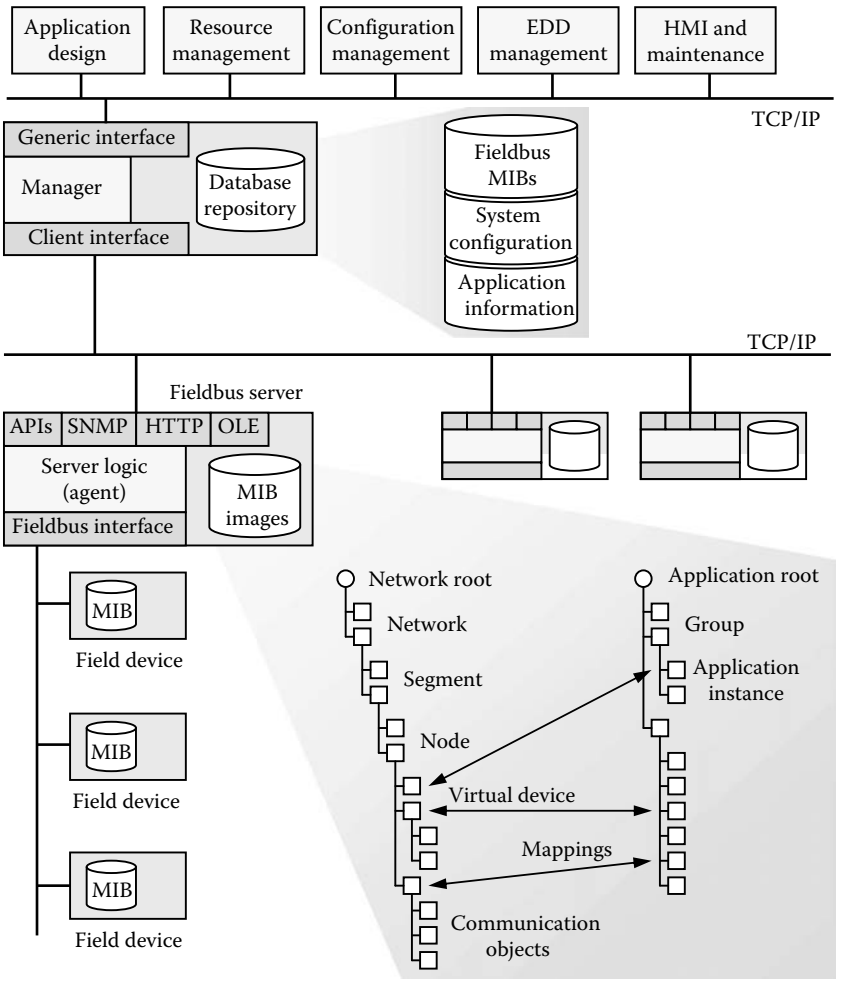


FIGURE 20.25 NOAH system architecture.

The NOAH system architecture shown in Figure 20.25 is a two-level manager/agent relation like the one suggested in Ref. [34], but more flexible. It consists of gateways (called fieldbus servers) that connect different fieldbus systems to a more general IP-based network and exchange data with a superordinate management application controlling the data base repository. External engineering or maintenance tools in turn access the data base repository manager to interact with the entire system. The core of the data model in NOAH relies on a management information base (MIB), a term borrowed from the ISO network management model which reminds of the primary intention of the NOAH project to tackle the management of fieldbus installations [35].

The MIB follows a clearly structure-oriented approach with the hierarchy levels: network—representing the individual fieldbus installations; segment—for further distinction inside larger installations; node—the individual devices; virtual device—distinct functions inside the device; communication objects—actual fieldbus objects. All objects have dedicated attributes specifying details. For a management application, usually more interested in functionality than structure, an alternative view on the MIB can be defined. Like in OPC, the application view comprises grouping and application instance objects; the two views are connected by appropriate mappings between application instances and virtual device objects.

Depending on the capabilities of the field devices, the actual implementation of the MIB may be distributed. As the MIB is structure-oriented, field devices can easily control their branch locally, and the fieldbus server needs to hold only a mirror of the data. Alternatively, the fieldbus server can act as a proxy and contain the full database. On the upper level, the data base repository summarizes all the fieldbus MIBs in the system. However, the pure fieldbus-related data are not sufficient to fully describe a plant, hence the repository also holds global data such as particular application information or overall system configuration, which can be accessed by appropriate tools.

Access to the fieldbus servers and the MIB data is possible through several interfaces to allow for easy integration in existing environments:

- Two native programming interfaces permit direct access to the MIB objects. These interfaces were defined in the predecessor project RACKS, which had the goal of specifying fieldbus-independent ways of accessing fieldbus data.
- SNMP suggests itself as a solution. Even though the NOAH MIB is not compliant with standard SNMP MIBs, a mapping is possible. Unlike SNMP, however, NOAH supports functions to manipulate the MIB at runtime. The proposed way to circumvent this problem in SNMP is to map the MIB manipulation functions to dedicated MIB variables, so that access to these variables triggers changes in the MIB. This idea was not tested in practice, and in view of the severe problems with SNMP MIBs concerning inflexibility and consistency [34], it is doubtful whether such an approach could work reliably.
- Object linking and embedding (OLE) was also a natural choice. By the time the decisions on the interfaces were taken, OPC had already begun to make its way. The adoption of two different views on the MIB objects clearly shows this influence.
- An HTTP-based interface permits easy access with Web browsers. The NOAH-specific information is coded in special tags inside the HTML files, and the read/write access to the actual objects is controlled via the URLs, which are specified in the communication with the HTTP server.

The common difficulty that a gateway can translate only data and basic services, but not more involved fieldbus-specific functions is solved in the usual way. Special functions can be activated by accessing dedicated MIB objects. In this case, the agent in the fieldbus server (which is fieldbus-specific in any case) has to execute the respective mechanism on the fieldbus.

After the project end in 2000, the results were submitted to CENELEC and IEC for standardization. The expectations after the successful termination of the research project were high. Nevertheless, the documents describing the research results first had to be transformed into a format useful for standardization which took time, and the work became entangled in tactical debates. Thus far, the device description has been the most successful outcome of NOAH: it was published as EN 50391 and IEC 601804-1, respectively. Work on the rest, especially the device profiles in part 2 of the IEC standard is going rather slowly.

20.6 Networking Networks—Interconnection in Heterogeneous Environments

Fieldbus systems were originally meant as part of a larger, comprehensive networking concept for automation. As the protocols were tailored to the needs of process control with its stringent real-time requirements and limited communication and computing resources, concepts from the higher layers of the networking hierarchy, notably the office area could not be used. Network interconnections were thus not straightforward. This situation is not restricted to fieldbus systems; it is common for all

networked embedded systems where communication systems were optimized in some respect and do not follow the mainstream networking technologies.

The question of how to interconnect different networks within the automation hierarchy is primarily an architectural choice and does not depend on the two (necessarily different) protocols used on both sides of the interconnection point. Against the backdrop of the ubiquity of the Internet technologies, however, the discussion and the development efforts today focus on the interconnection of fieldbus systems and IP-based networks in whatever form. From an architectural point of view, there are two main possibilities to achieve a fieldbus/Internet interconnection, both of which are being used in practice:

- Tunneling or encapsulation of one protocol in the other, either in a strictly vertical sense or to achieve a connection between remote fieldbus segments via a higher-level backbone network
- Providing protocol, service, and data translation via a gateway

The topological view of both approaches is identical. In both bases, there is a central node linking the two networks. Usually, the term "gateway" is used for such an interconnection node. In the following, we will prefer the more neutral term "access point" (not to be confused with the access points used in wireless networks) to avoid misconceptions regarding the functionality of the device, which differs significantly between gateway and tunneling approach.

20.6.1 Protocol Tunneling

Tunneling in connection with communication networks essentially means that data frames of one protocol are wrapped into the payload data of another protocol without any modification or translation. Thus, two possibilities exist: the fieldbus protocol can be encapsulated in the Internet Protocol on the higher level, or a protocol of from the IP suite can be passed over the fieldbus. Tunneling is well-known in the office world as a means to set up virtual private networks for secure communication of, e.g., remote offices. Although tunnels in this area normally operated on the data link layer (like the Layer 2 Tunneling Protocol L2TP), in fact any PDU of any protocol layer can be tunneled over any other; there is no compelling reason to restrict tunneling to the data link layer.

20.6.1.1 IP Tunneling over Fieldbus Protocols

In the automation area, the currently more common tunneling approach is to encapsulate IP packets in fieldbus messages. This is a direct consequence of the widespread use of Web-based interfaces for engineering tools and opens a channel for the upper layer protocols required for, e.g., direct Web access to the field devices [54]. While in most cases not foreseen in the beginning, this possibility has recently been included in several fieldbus systems [36,59]. At first glance, IP tunneling provides an easy way to achieve integration, as Figure 20.26 shows. The field devices run an IP-based service such as a Web server providing data for a corresponding application in the Internet, which can directly access the device or process data. While the typically limited computing resources at the field device used to be a counter-argument for this solution for a long time, the availability of embedded controllers with lightweight (maybe even on-chip) TCP/IP stack implementations [37] and Web servers for smart devices solves the problem [38–40]. As these devices usually also have integrated Ethernet controllers, the question arises why additional fieldbus interfaces should be added instead of using Ethernet directly on the field level.

Another essential design consideration for the access point is the way the traffic is being handled. This has two facets: addressing and scheduling. The addressing problem becomes obvious if we consider that the "tunnel" is not just a single communication path connecting two nodes, but has several exits. Therefore, the access point must translate between the IP address used to identify the field

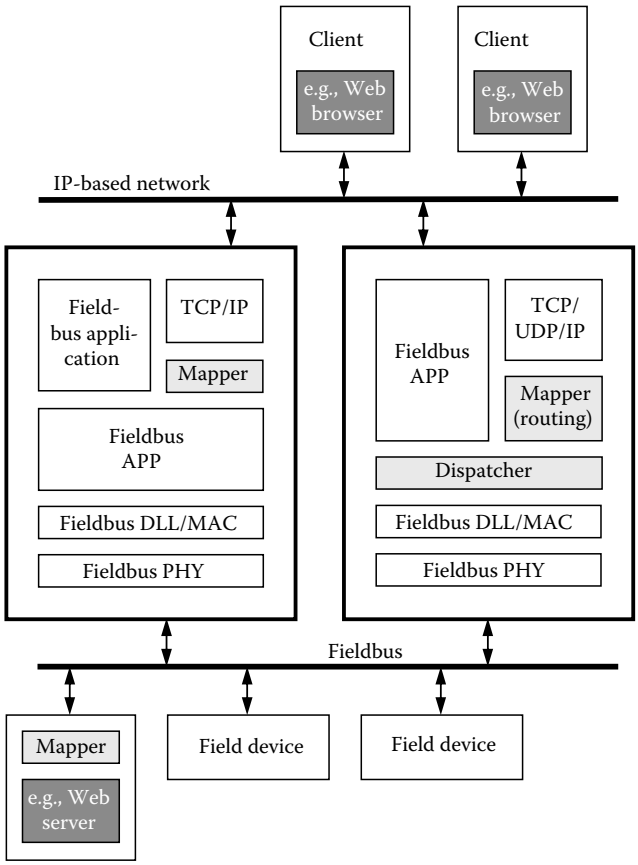


FIGURE 20.26 Protocol and software structure for IP tunneling over a fieldbus.

device and the fieldbus address used to actually communicate with the device over the fieldbus. This requires the implementation of an IP routing service on the access point. Additionally, a network address translation including the mapping of services to dedicated ports might be necessary if the IP address ranges on either side of the access point are not consistent, e.g., if private addresses are used for the fieldbus and public ones for the outside network. The address translation problem can be alleviated by extended addressing schemes allowing the use of IP-like addresses directly on the fieldbus. For example, PROFIBUS supports such an extended addressing mode.

The second aspect of traffic handling is scheduling. IP packets tend to be much larger than fieldbus data frames, which were optimized for the timely transmission of small pieces of data. Sending IP packets over a fieldbus therefore demands fragmentation and the introduction of fragment numbers to permit a correct reassembly. Contrary to statements frequently found in the literature [36,41], there is no lower limit for the fragment size. The argument that the IP header must not be fragmented (thereby introducing a minimum fragment size of 20 bytes) is valid only if the native fragmentation mechanisms "inside" IP are used. This is, however, not a stringent requirement for the implementation of a tunneling solution, and the IP packet can be treated like any arbitrary binary large object and cut into pieces of arbitrary size. What is necessary, though, is to properly schedule the insertion of the IP data stream in the usual fieldbus traffic to avoid possible distortion of the real-time characteristics. These additional functionalities require a separate mapping and dispatching layer in all devices participating in the IP communication channel, which could be placed on top of either the

data link layer [41,42] or the application layer (in the way safety profiles introduce additional protocol features like transaction numbering to safeguard the communication).

Another fundamental problem with tunneling concerns the response time of the IP channel established over the fieldbus. IP is basically a peer-to-peer protocol, which means that any node must in principle be able to initiate a communication at any time. In particular in master–slave fieldbus systems, this cannot be guaranteed easily because slave nodes cannot become active without being told so by the master. If in addition the fieldbus is slow, the latency introduced by the polling cycle might lead to troubles with (often hard-coded) timeouts in the protocol stacks used on the nodes in the IP-based network. The fact that data transfer between a fieldbus and an IP-based network is normally limited to client–server communication does not help either. Typically, the field device provides the server and the client is, e.g., a configuration tool in the outside world. One could expect that in this case, excessive polling latencies do not occur—provided, of course, that the access point is also the fieldbus master. Unfortunately, if TCP is used as transport protocol (like in the popular applications of embedded Web servers), the delivery of requests and responses is inherently asynchronous. Things get worse if the request and response packets exceed the maximum size of the fieldbus payload and must be fragmented. The additional traffic needed to transfer all fragments will expand over subsequent polling cycles, further deteriorating the end-to-end delay in the IP channel. For a reasonably efficient tunneling of IP over a slow master–slave fieldbus, it would be necessary to influence the polling cycle directly from the TCP stack, which is not a viable solution. Consequently, tunneling is possible only in multi-master fieldbusses or in sufficiently "fast" master/slave systems.

Despite the conceptual problems, tunneling approaches are available even for master–slave fieldbusses. Within the scope of the R-Fieldbus project, the feasibility of IP over PROFIBUS to include multimedia data streams was demonstrated [41]. Apart from this research project, IP tunneling solutions have been developed for Interbus and WorldFIP. Interbus is an extreme example, because the IP traffic is transported in a communication channel similar to the parameter channel [44,59]. This channel has only a small capacity of, say, eight bytes per cycle in order not to interfere with the real-time process data, and one byte is needed for control purposes. Depending on the size of the network and the selected baud rate, one cycle may take a few milliseconds. As an IP packet usually is of the size of several hundred bytes, it is evident that the performance of the IP channel is rather limited. In WorldFIP, the situation is better. IP packets are transmitted in message frames outside the time-critical section of the bus cycle. The messages can be up to 256 bytes long and are thus large enough to avoid excessive fragmentation [36]. Together with a high raw data rate on the fieldbus itself, this offers a reasonable IP channel that is being used in practical applications, e.g., for embedded Web servers inside field devices [65].

20.6.1.2 Fieldbus Data Tunneling over Backbone Networks

The tunneling variant where fieldbus messages are encapsulated into IP-based protocols and processed at a distant node is less relevant in practice as far as actual implementations are concerned. Nevertheless, there are applications. One is to connect remote fieldbus segments over a backbone network. A wide field of application for this concept are network-based control systems, for instance in building automation [73,74]. Like in the reverse case, the latency introduced by the tunnel itself and the protocol handling at the endpoints matters. It is therefore not astonishing that research work was devoted to solutions involving ATM as backbone. ATM offers sophisticated QoS mechanisms which ensure bounded transmission delays and therefore are basically able to retain real-time characteristics of the fieldbus. Implementations were developed, e.g., for PROFIBUS [75,76], but without achieving any practical relevance.

No matter if ATM or general IP-based networks (possibly also using QoS features if available) are used as backbone network, a really transparent interconnection where the tunnel does not noticeably impair the timing of the fieldbus network is hardly possible. It would theoretically be possible only if

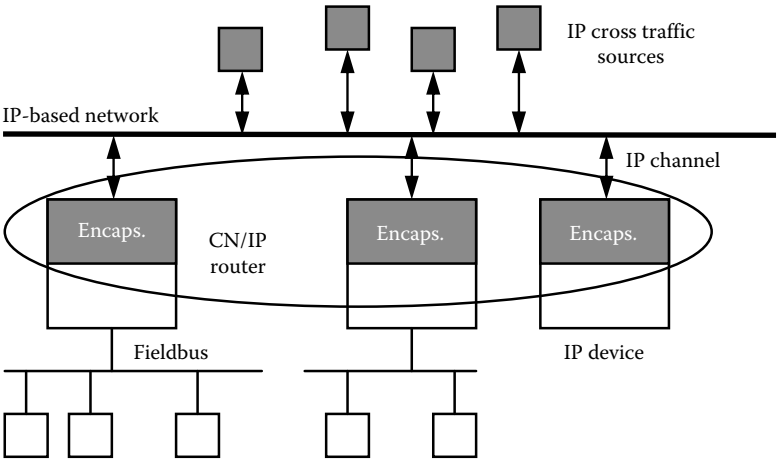


FIGURE 20.27 Structure of an IP channel for fieldbus data tunneling over the Internet.

the transmission inside the tunnel is much faster than in the fieldbus. As this is not normally the case, the tunnel endpoints rather act as remote bridges separating the two segments (in the sense of two timing domains). To that end, the fieldbus must either support multiple segments and thus some sort of routing, or traffic forwarding from one segment to the other needs to be introduced "manually" on a higher protocol layer (or a respective application).

In recent years, the EIA-852 standard has been established to cope with some of these problems [74,85]. It introduces means to define and manage IP channels for a peer-to-peer transmission of fieldbus data packets. Basically, the methods defined in the standard are generic. In practice, implementations are currently only defined for EIA-709 (LonWorks) and EIA-600 (CEBus). It is not surprising that the standard evolved in the building automation domain. In this environment, large IT networks are common as well as localized control networks which might benefit from an interconnection over IP. Consequently, the devices in an EIA-852 system can either be purely IP-based or regular control network devices which are connected to the IP network through the access points (in EIA-852 terminology called control network/IP tunneling routers). These devices connected over the IP-based network form together the IP channel (Figure 20.27). The native PDUs are encapsulated in UDP frames and routed to the respective recipients on the IP channel. The tunneling routers take care of appropriate data flow control mechanisms. Sequence numbers are assigned to the packets to allow for correct reassembly of the packet order even if the packets used different paths to reach the receiver. If the receiver recognizes that packets are missing, it can—depending on the configuration—either continue and drop them when they finally arrive (late packets are useless in control applications) or store the subsequent for a given time (the escrow time in EIA-852 terminology), hoping that the missing packet(s) still arrive. To increase the efficiency of the fieldbus packet encapsulation in the UDP frames, there is the possibility to bundle subsequent packets into one single UDP frame (packet bunching). This introduces extra delay, but reduces the encapsulation overhead (especially the UDP, IP, and media-specific headers). To cope with excessive transmission delays on the IP channel, timestamps can be issued for the frames to measure the delay. Frames exceeding a predefined transmission time will then be discarded as stale packets.

20.6.2 Gateways

The alternative approach to implement a connection between a fieldbus and an IP-based network is to design the central access point as a gateway. Like with the tunneling approach, the gateway is a full member of the fieldbus on the one side and can be accessed through IP-based mechanisms via the

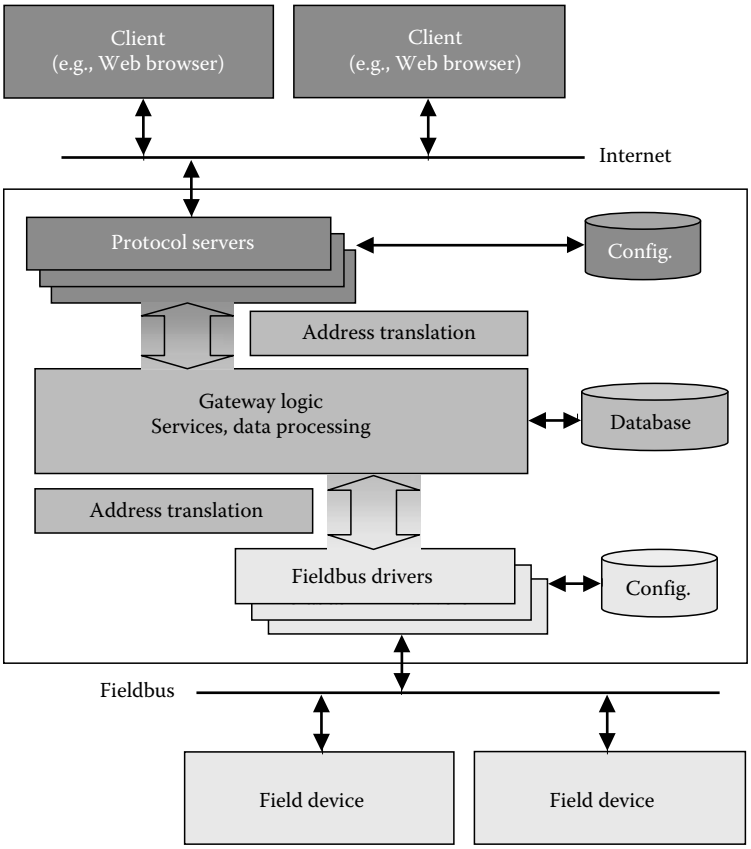


FIGURE 20.28 Network topology for a gateway-based interconnection and general architecture of a gateway.

Internet (Figure 20.28). What is different, though, is the way the information exchange is handled. In the IP tunneling approach, the client connects directly to the server running on the field device. The access point just forwards the IP traffic, but is not further involved in the data transfer. In the gateway approach, the access point takes the role of a proxy representing the fieldbus and its data to the outside world. It fetches the data from the field devices using the usual fieldbus communication methods and is the communication partner addressed by the client.

There is no best or most suitable architecture for a gateway. It can act as an abstract interface to an automation system, independent of both the fieldbus protocol (because it operates on application level) and the fieldbus-specific coding of the data. This way, it can provide unified access to even more than one fieldbus simultaneously. The software structure of the gateway implementation can reflect this demand for versatility and exhibit a modular structure [52,87]. The three-level approach shown in Figure 20.28 is one possibility to achieve modularity. Alternatively, the architecture can as well be monolithic and tailored to one specific case. This high degree of flexibility may be the reason that gateways are the preferred interconnection concept in practice, especially for remote monitoring applications [88,89]. Another reason is that gateways are an ideal point to implement access control [90,108,109]. In Web-based integration scenarios, the gateway is usually described and implemented as a Web portal [91,92].

Unlike the tunneling approach, where each field device has to provide and maintain its data on an individual basis, information processing for the client in the Internet can be centralized. This enables the gateway to set up a comprehensive and consistent view of the automation system, a kind

of "process image" that can be viewed from outside. While the basic design of access points in the tunneling case does not allow many degrees of freedom, the operations performed by a gateway are on application level and can thus be much more versatile (see Ref. [93] for a discussion of design considerations). Services offered by the gateway can include autonomous data-processing features supporting low-level SCADA functionalities (like threshold monitoring), automatic update of the list of connected nodes and data points [100,107], or notifications of a client in case predefined events occur.

A large variety of possibilities exist with regard to the way data points can be represented by the gateway. If specific profiles are available for the field devices, they can be used [47,107]. Another way is to simply provide a list of data points, organized either following a function-oriented approach or a structure-oriented one based on the topology of the fieldbus and its nodes [110]. While the structure of the database holding the data inside the gateway is naturally fully free, the way of accessing the data from outside, i.e., from the IP-based network, may impose stringent limitations on the organization of the data because the protocol used must support the data structures implemented on the gateway.

On the lower side of the gateway, the connection to the fieldbus provides again many degrees of freedom in that there is no general guideline on how to implement it. The fieldbus driver has to be developed anew for each case and has to exploit the data transfer mechanisms supported by the fieldbus. This may also affect the overall traffic flow handling implemented on the gateway. One possibility is to forward incoming requests to the respective field device and wait for the response to send it back to the client. The alternative is to use a caching mechanism and to answer requests with data point values from the cache, accompanied by a timestamp indicating the age of the value [52]. Which strategy should be preferred depends on the capabilities of the driver interface as well as the desired scope of data to be handled by the gateway, i.e., whether the gateway needs to have access to process or configuration data.

With regard to possible real-time applications in automation, the situation is considerably different to the tunneling approach. The protocol conversion in the gateway necessarily terminates QoS features on either side because the translation between the two networks is accomplished on application level. This is the purpose of a gateway: both the fieldbus and the IP-based network are able to operate asynchronously, the gateway serves as buffer in between. Unless the gateway has means to synchronize the data transfer on both sides (e.g., by acting as master on the fieldbus), no end-to-end timing guarantees can be given, even if both networks do have real-time capabilities, because the protocol conversion does not allow for an end-to-end communication relation across the gateway. Consequently, gateways are typically used for process monitoring or management purposes, where stringent real-time requirements do not exist.

For the sake of completeness, it must be noted that gateways can also "horizontally" interconnect different fieldbus systems. Although appealing in principle, this topic has never reached practical relevant because of the extreme heterogeneity of the fieldbus solutions which prohibits suitable one-to-one mapping of data and services. One approach introduces an interoperability layer based on the User Layer philosophy providing a common object model and messaging domain for distributed applications [111]. The fieldbus drivers are implemented with OPC. Another experimental solution employs function blocks according to IEC 61499 to provide the necessary abstraction layer between the fieldbusses [127].

20.7 Industrial Ethernet—The New Fieldbus

As stated before, Ethernet has become increasingly popular in automation. And like in the early days of fieldbus systems, this boom is driven mainly by the industry—on an academic level, the use of Ethernet had been discussed decades ago. At that time, Ethernet was considered inappropriate because of its lack of real-time capabilities. With the introduction of switched Ethernet and certain modifications of the protocol, however, these problems have been alleviated. And even if there are

still doubts about the predictability of Ethernet [55], its penetration into the real-time domain will influence the use of fieldbus-based devices and most likely restrict the future use of fieldbus concepts [56,57]. Today, Ethernet already takes the place of mid-level fieldbus systems, e.g., for the connection of PLCs.

Using Ethernet on the field level, one first of all has to overcome the problem of the inherent lack of determinism. For Ethernet as shared medium, and if no hard real-time behavior is requested, various types of traffic smoothing techniques have been proposed [128,129], where real-time packets are given priority over non-real-time ones. The goal is to eliminate contention within each local node, and to shape non-real-time traffic to reduce the chance for collision with real-time packets from the other nodes. The introduction of Switched Ethernet further alleviated the problem [130]. Much research work consequently focused on how to reduce the queuing delays inside the switches [131,132] and on traffic smoothing methods for Switched Ethernet [133].

One of the main arguments used to promote Ethernet on the field level is that because it is the same network technology as in the office world, a straightforward integration is possible, i.e., both automation and office domain can in principle be connected to one single enterprise network. A quick look at reality, however, shows that things are different. Ethernet per se is but a solution for the two lower OSI layers, and as fieldbus history already showed, this is not sufficient. Even if the commonly used Internet protocol suite with TCP/UDP/IP is taken into account, only the lower four layers are covered. Consequently, there are several possibilities to get Ethernet or Internet technologies into the domain currently occupied by fieldbus systems:

- Tunneling of a fieldbus protocol over UDP/TCP/IP
- Tunneling of TCP/IP over an existing fieldbus
- Definition of new real-time-enabled protocols
- Limitation of the free medium access in standard Ethernet

All of these possibilities are actually used in practice (Figure 20.29). In the beginning, all research work carefully avoided any concepts violating the Ethernet standards. Compatibility and conformity

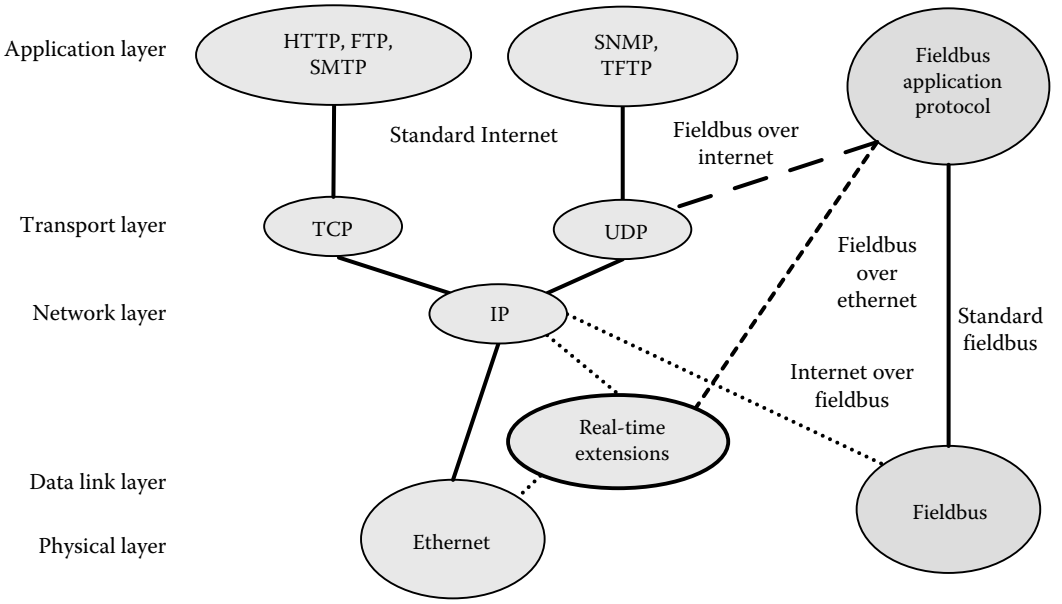


FIGURE 20.29 Structures of Ethernet and fieldbus combinations.

was the prime goal. Especially those approaches already contained in the IEC 61158 standard employ existing fieldbus application layer protocols on top of IP-based transport mechanisms (TCP or UDP, respectively, depending on the services needed) that replace the lower fieldbus layers [31]. In fact, this is closely related to the fieldbus-over-Internet tunneling strategy discussed in the previous section. The following four examples pursue this approach:

- High Speed Ethernet variant of Foundation Fieldbus is an application of the existing Fieldbus Foundation's H1 protocol wrapped in UDP/IP packets [134].
- Ethernet/IP (IP in this case standing for Industrial Protocol) uses the Control and Information Protocol already known from ControlNet and DeviceNet [58]. This application layer protocol is sent over TCP or UDP, depending on whether configuration or process data have to be transmitted.
- Modbus/TCP [68] is based on standard Modbus frames encapsulated in TCP frames. For more stringent real-time requirements, a real-time variant of the publisher–subscriber model was developed by IDA Group (Interface for Distributed Automation, now merged with Modbus Organization), which builds on UDP.
- P-NET on IP [135] defines a way to wrap P-NET messages in UDP packets.

These Industrial Ethernet solutions build on Ethernet in its original form, i.e., they use the physical and data link layer of ISO/IEC 8802-3 without any modifications. Furthermore, they assume that Ethernet is low loaded or fast Ethernet switching technology is used to get a predictable performance. Switching technology does eliminate collisions, but delays inside the switches and lost packages under heavy load conditions are unavoidable also with switches [60]. This gets worse if switches are used in a multilevel hierarchy, and may result in grossly varying communication delays. The real-time capabilities of native Ethernet are therefore limited and must rely on application-level mechanisms controlling the data throughput. For advanced requirements, like drive controls, this is not sufficient. These known limitations of conventional Ethernet stimulated the development of several alternative solutions that were more than just adaptations of ordinary fieldbus systems. These entirely new approaches were originally outside the IEC standardization process, but are now candidates for inclusion in the RT Ethernet standard, i.e., the second volume of IEC 61784.

The initial and boundary conditions for the standardization work, which started in 2003, are targeted at backward compatibility with existing standards. First of all, real-time Ethernet (RTE) is seen as an extension to the Industrial Ethernet solutions already defined in the communication profile families in IEC 61784-1. Furthermore, coexistence with conventional Ethernet is intended. The scope of the working document [61] states that "the RTE shall not change the overall behavior of an ISO/IEC 8802-3 communication network and their related network components or IEEE 1588, but amend those widely used standards for RTE behaviors. Regular ISO/IEC 8802-3-based applications shall be able to run in parallel to RTE in the same network."

The work program of the RTE working group essentially consists of the definition of a classification scheme with RTE performance classes based on actual application requirements [57,63]. This is a response to market needs which demand scalable solutions for different application domains. One possible classification structure could be based on the reaction time of typical applications:

- First low speed class with reaction times around 100 ms. This timing requirement is typical for the case of humans involved in the system observation (10 pictures per second can already be seen as a low-quality movie), for engineering, and for process monitoring. Most processes in process automation and building control fall into this class. This requirement may be fulfilled with a standard system with TCP/IP communication channel without many problems.
- In a second class, the requirement is a reaction time below 10 ms. This is the requirement for most tooling machine control systems like PLCs- or PC-based control. To reach this

timing behavior, special care has to be taken in the RTE equipment. Sufficient computing resources are needed to handle the TCP/IP protocol in real-time or the protocol stack must be simplified and reduced to get these reaction times on simple, cheap resources.

- Third and most demanding class is defined by the requirements of motion control: To synchronize several axes over a network, a time precision well below 1 ms is needed. Current approaches to reach this goal rely on modifications of both protocol medium access and hardware structure of the controllers.

During the last few years, a number of industrial solutions appeared that tackled the real-time requirements, mostly on the basis of switched Ethernet. Still, as with fieldbus systems, they were tailored to specific needs. Not even the use of standard Ethernet is really a common denominator, and above the data link layer, the approaches are completely different. Some use standard TCP/UDP/IP mechanisms for transmitting data, maybe enhanced by additional software layers to support both real-time and non-real-time communication, and some use dedicated communication stacks that bypass the entire IP suite. Figure 20.30 sketches the various appearances of the protocol stack. Manifold differences are also possible on the physical layer. Some approaches foresee redundant media (VNET/IP, TCnet), PROFINET-I/O uses dedicated built-in switches to reduce the data transmission jitter [64], and EtherCAT [66] as well as SERCOS III [46] need dedicated controllers. Ethernet Powerlink uses the old shared Ethernet and places a master–slave scheduling system on top of it [67]. Common to many proposed networks is that they employ clock synchronization to support real-time applications. To this end, the recent standard IEEE 1588 [62], which originally emerged in the instrumentation area, was officially adopted also by IEC. The specific requirements in the automation domain have led to several suggestions for improvement of the standard [136,137], and work on the next revision was already started.

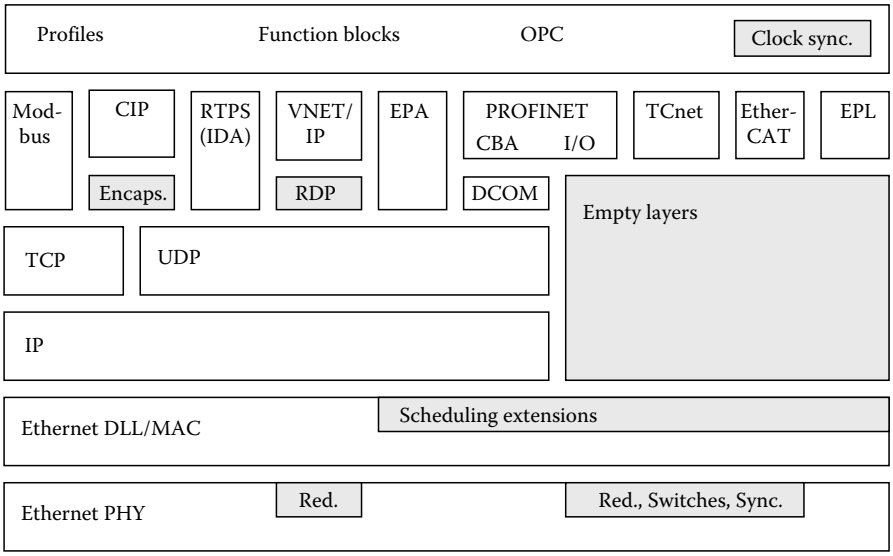


FIGURE 20.30 Protocol architecture of selected RTE solutions proposed for IEC 61784-2.

Given the multitude of approaches, despite the early hope that Ethernet could be the basis for a unique industrial communication solution, the situation has not too much changed compared to the heterogeneity of the traditional fieldbus systems. Interoperability between different Industrial Ethernet solutions is not possible in a direct way and must be established on a higher level by means of profiles (which is actually done to allow for cooperation between "old" fieldbusses and "new" Ethernet installations) or middleware layers like OPC. The only conceivable improvement compared to fieldbus technology is that even with all proprietary modifications, Ethernet and to a large extent also the IP suite are being recognized as technological basis for the new generation of industrial communication systems. And with a view on vertical integration, the main benefit of Industrial Ethernet is that all approaches allow for a standard TCP/UDP/IP communication channel in parallel to fieldbus communication. Even the RTE solutions (like PROFINET, Ethernet Powerlink, EtherCAT, etc.) have such a conventional channel for configuration purposes. This is related to the IP-over-fieldbus tunneling concepts discussed earlier, but unlike the classical fieldbus systems, where such IP tunnels were introduced long after the fieldbus development and thus often had to cope with performance problems, they are now an integral part of the system concept right from the beginning on.

The separation of real-time and non-real-time traffic is accomplished on Ethernet MAC level with prioritization or TDMA schemes together with appropriate bandwidth allocation strategies. In such a parallel two-stack model, IP channels are no longer stepchildren of industrial communication, but offer sufficient performance to be used for regular data transfer. While this enables in principle the coexistence of automation and nonautomation applications on Industrial Ethernet segments, the mixing of automation and office is not advisable for performance, but more important for security reasons. The value of this standard IP channel is rather to be seen in a simple direct access path to the field devices. Therefore, the currently favored solutions for configuration tools (i.e., XML, SOAP, and more generally Web technology) can be used consistently. This again does not mean that Industrial Ethernet solutions are interoperable or use the same configuration tools, but at least the basic principles are the same.

Actually, all this could have already been done with traditional fieldbus systems as well, and it certainly would have been done had especially the achievements of the Internet and the WWW been available in the early 1980s. So, what we see today with the rapid evolution of Ethernet in automation can in fact be regarded as a second wave of fieldbus development which takes into account all the technological achievements of the last decade and exploits them for the field level, thereby making particularly vertical integration significantly easier.

20.8 Aspects for Future Evolution

Fieldbus systems have come a long way from the very first attempts of industrial networking to contemporary highly specialized automation networks. What is currently at hand—even after the selection process during the last decade—nearly fully covers the complete spectrum of possible applications. Nevertheless, there is enough evolution potential left [46,70,86]. On the technological side, the communication medium itself allows further innovations. The currently most promising research field for technological evolution is the wireless domain. The benefits are obvious: no failure-prone and costly cabling and high flexibility, even mobility. The problems on the other hand are also obvious: very peculiar properties of the wireless communication channel must be dealt with, such as attenuation, fading, multipath reception, temporarily hidden nodes, the simple access for intruders, and many more [71]. Wireless communication options do exist today for several fieldbusses [72]. Up to now, they have been used just to replace the conventional data cable. A really efficient use of wireless communication, however, would necessitate an entire redefinition of at least the lower fieldbus protocol layers. Evaluation of currently available wireless technologies from the computer world with respect to their applicability in automation is a first step in this direction. Ultimately we can

expect completely new automation networks optimized for wireless communication, where maybe only the application layer protocol remains compatible with traditional wired solutions to achieve integration.

Driven by ever more demanding application areas, communication networks for safety-relevant systems gain importance. In special fields such as x-by-wire for vehicles and avionics, dedicated fieldbus systems were developed that specifically addressed the problem of reliable communication [83]. Examples are TTP, FlexRay, byteflight, or ARINC 629. These networks are relatively new and can build on substantial experience of the scientific community as well as industry with other fieldbus systems. As this domain is subject to very stringent normative regulations and thus very conservative, it was dominated for a long time (and still is) by point-to-point connections between devices. The first "older" fieldbus system to penetrate this field was the CAN-based Safety Bus [77]. It took a long time and effort for this system to pass the costly certification procedures. Nevertheless, it was finally accepted also by the users, which was by no means obvious in an area concerned with the protection of human life, given that computer networks usually have the psychological disadvantage of being considered unreliable. After this pioneering work, other approaches like the ProfiSafe profile [78], INTERBUS Safety [79], AS-i Safety [80], and recently also Ethernet/IPsafety [81], WorldFIP [82], and SafetyLON (still under development at the time of writing) readily followed. All these traditional fieldbus systems that were originally not designed for safety-critical applications implement this particular functionality by means of dedicated profiles. In practice, they wrap additional safety protocols into the normal payload data which foresee a set of measures to make communication more reliable: sequence numbers, additional CRCs and confirmations, timestamps, heartbeat functions and timeouts together with safety monitors and built-in test functions for the hardware components detect residual errors and may typically achieve safety integrity level 3 according to IEC 61508, which makes them applicable for fire emergencies, alarming, traffic and transportation applications, or power plant control.

Apart from communication protocol issues, there are two major trends to be noticed. One is the growing complexity of networks and networked systems in general. This is manifested by the increasing integration of fieldbus systems in higher-level, heterogeneous networks, and process control systems on the one hand and the massive use of Internet technologies to achieve simplification and possible harmonization of existing solutions. This in fact drastically reduces the number of layers in the traditional network hierarchy, which only reflects the trend toward peer-to-peer networking on a protocol level. The other trend is the still increasing capabilities of embedded devices, the possibility to integrate more computational resources while at the same time reducing energy consumption. Systems on Chip with on-chip memory, network interfaces, and the computing power of a complete industrial PC offer sufficient resources for smart and low-cost sensors and actuators. This evolution is on the one hand the foundation of the current boom of Ethernet in automation. On the other, it will stimulate more research in the already booming field of sensor networks [84]. What is likely to be seen in the future are much more ubiquitous Ethernet- and Internet-based concepts, probably optimized to meet special performance requirements on the field level but still compatible with the standards in the management area. At any rate, these concepts and protocols will have to be scalable to allow for seamless integration of low-level, highly specialized sensor/actuator networks tailored meeting the demands of low power consumption, small-footprint implementation, high flexibility, and self-organization. The fieldbus of the future will be a very versatile thing: embedded network and a network of embedded systems alike.

20.9 Appendix

The tables presented here give an overview of selected fieldbus systems, categorized by application domain (Tables 20.2 through 20.5). The list is necessarily incomplete, although care has been taken to

TABLE 20.2 Instrumentation and PCB-Level Busses

Fieldbus	Developer (Country)	Introduced in	Standard	Refs.
CAMAC	ESONE (Europe)	1969 (start of development 1966)	IEEE 583 (1970, 1982, 1994) IEEE 595 (1974, 1982) IEEE 596 (1972, 1982) IEEE 758 (1979)	[112]
GPIB (HP-IB)	Hewlett-Packard (USA)	1974 (start of development 1965)	ANSI IEEE-488 (1975, 1978) ANSI IEEE-488.2 (1987, 1992) IEC 60625 (1979, 1993)	[113–115]
HP-IL	Hewlett-Packard (USA)	1980 (start of development 1976)	—	[18]
I^2C	Philips (the Netherlands)	1981	—	[116]
M-Bus	University of Paderborn, TI, Techem (Germany)	1992	EN 1434-3 (1997)	[117]
Measurement bus	Industry consortium (Germany)	1988	DIN 66348-2 (1989) DIN 66348-3 (1996)	

Source: Sauter, T., in *The Industrial Communication Handbook*, CRC Press, Boca Raton, FL, 2005, 7.1–7.39. (With permission.)

TABLE 20.3 Automotive and Aircraft Fieldbusses

Fieldbus	Developer (Country)	Introduced in	Standard	Refs.
ABUS	Volkswagen (Germany)	1987	—	[118]
ARINC	Aeronautical Radio, Inc. (USA)	1978	AEEC ARINC 429 (1978, 1995) AEEC ARINC 629 (1989)	[141]
Byteflight	BMW and industry consortium	2000 (start of development 1996)	Open spec.	
CAN	Bosch (Germany)	1986 (start of development 1983), CAL 1992	ISO 11898 (1993, 1995) ISO 11519 (1994)	[119]
FlexRay	DaimlerChrysler, BMW (Germany)	2002	—	
J1850	Ford, GM, Chrysler (USA)	1987	SAE J1850 (1994, 2001) ISO 11519-4	[118]
J1939	SAE (USA)	1994	SAE J1939 (1998)	[118]
LIN	Industry consortium	1999	Open spec.	
MIL 1533	SAE (military and industry consortium, USA)	1970 (start of development 1968)	MIL-STD-1553 (1973) MIL-STD-1553A (1975) MIL-STD-1553B (1978)	[120]
SwiftNet	Ship Star Assoc., Boeing (USA)	1997	IEC 61158 (2000, dropped 2008)	
TTP	Vienna University of Technology (Austria)	1996	—	[118]
VAN	Renault, PSA Peugeot-Citroen (France), ISO TC22	1988	ISO 11519-3 (1994)	[118]

TABLE 20.4 Fieldbusses for Industrial and Process Automation and Their Foundations

Fieldbus	Developer (Country)	Introduced in	Standard	Refs.
ARCNET	Datapoint (USA)	1977	ANSI ATA 878 (1999)	[121]
ASi	Industry and university consortium (Germany)	1991	EN 50295-2 (1998, 2002) IEC 62026-2 (2000)	
Bitbus	Intel (USA)	1983	ANSI IEEE 1118 (1990)	
CC-Link	Mitsubishi (Japan)	1996	Open spec.	
CANopen	CAN in Automation (user group, Germany)	1995 (start of development 1993)	EN 50325-4 (2002)	[119]
ControlNet	Allen-Bradley (USA)	1996	EN 50170-A3 (2000)	[121]
DeviceNet	Allen-Bradley (USA)	1994	EN 50325-2 (2000)	[119]
FF	Fieldbus Foundation (industry consortium, USA)	1995 (start of development 1994)	BSI DD 238 (1996) EN 50170-A1 (2000)	[121]
Hart	Rosemount (USA)	1986	Open spec.	
Interbus-S	Phoenix Contact (Germany)	1987 (start of development 1983)	DIN 19258 (1993) EN 50254-2 (1998)	[122]

(*continued*)

TABLE 20.4 (continued)

Fieldbus	Developer (Country)	Introduced In	Standard	Refs.
MAP	General Motors (USA)	1982 (start of development 1980)	MAP 1.0 (1982) MAP 2.0 (1985) MAP 3.0 (1988)	[123]
MMS	ISO TC 184	1986	ISO/IEC 9506 (1988, 2000)	
Modbus	Gould, Modicon (USA)	1979	Open spec.	
PDV-Bus	Industry and university consortium (Germany)	1979 (start of development 1972)	DIN 19241 (1982)	[115,124]
P-NET	Process Data (Denmark)	1983	DS 21906 (1990) EN 50170-1 (1996)	
PROWAY C	IEC TC 65	1986 (start of development 1975)	ISA S72.01 (1985) IEC 60955 (1989)	[14]
PROFIBUS	Industry and university consortium (Germany)	1989 (start of development 1984)	FMS: DIN 19245-1-2 (1991) DP: DIN 19245-3 (1993) PA: DIN 19245-4 (1995) FMS/DP: EN 50170-2 (1996) DP: EN 50254-3 (1998) PA: EN 50170-A2 (2000)	
SDS	Honeywell (USA)	1994	EN 50325-3 (2000)	[119]
Sercos	Industry consortium (Germany)	1989 (start of development 1986)	IEC 61491 (1995) EN 61491 (1998)	
Seriplex	APC, Inc. (USA)	1990	IEC 62026-6 (2000)	[121]
SINEC L2	Siemens (Germany)	1992	—	
SP50 Fieldbus	ISA SP 50 (USA)	1993	ISA SP 50 (1993)	
(World) FIP	Industry and university consortium (France)	1987 (start of development 1982)	AFNOR NF C46601-7 (1989–1992) EN 50170-3 (1996) DWF: AFNOR NF C46638 (1996) DWF: EN 50254-4 (1998)	[16]

Source: Sauter, T., in *The Industrial Communication Handbook*, CRC Press, Boca Raton, FL, 2005, 7.1–7.39. (With permission.)

TABLE 20.5 Fieldbusses for Building and Home Automation

Fieldbus	Developer (Country)	Introduced In	Standard	Refs.
BACnet	ASHRAE SPC135P (industry consortium, USA)	1991	ANSI/ASHRAE 135 (1995) ENV 1805-1 (1998) ENV 13321-1 (1999) ISO 16484-5 (2003)	
Batibus	Industry consortium (France)	1987	AFNOR NF 46621-3, 9 (1991) ENV 13154-2 (1998)	
CEBus	Industry consortium (USA)	1984	ANSI EIA 600 (1992)	
EHS	Industry consortium (Europe)	1987	ENV 13154-2 (1998)	
EIB	Industry consortium (Germany)	1990	AFNOR NFC 46624-8 (1991) DIN V VDE 0829 (1992) ENV 13154-2 (1998)	[125]
HBS	Industry consortium (Japan)	1986 (start of development 1981)	EIAJ/REEA ET2101	
LonWorks	Echelon (USA)	1991	ANSI EIA 709 (1999) ENV 13154-2 (1998)	[121,126]
Sigma I	ABB (Germany)	1983	—	
X10	Pico Electronics (UK)	1978 (start of development 1975)	—	[126]

Source: Sauter, T., in *The Industrial Communication Handbook*, CRC Press, Boca Raton, FL, 2005, 7.1–7.39. (With permission.)

include all approaches that either exerted a substantial influence on the evolution of the entire field or are significant still today. The year of introduction refers to the public availability of the specification or first products. This year is also the one used in the timeline in Figure 20.3. Note that despite careful research, the information obtained from various sources was frequently inconsistent, so that there may be an uncertainty in the figures. Where respective data could be obtained, the start of the project has been listed as well because there are several cases where the development of the fieldbus took a long time before the first release.

References

1. International Electrotechnical Commission, IEC 61158, Digital data communications for measurement and control—Fieldbus for use in industrial control systems, 2003.
2. Fieldbus Foundation, *What is Fieldbus?* http://www.fieldbus.org/About/FoundationTech/
3. G. G. Wood, Fieldbus status 1995, *Computing & Control Engineering Journal*, 6, December 1995, 251–253.
4. G. G. Wood, Survey of LANs and standards, *Computer Standards & Interfaces*, 6, 1987, 27–36.
5. N.P. Mahalik (ed.), *Fieldbus technology: Industrial Network Standards for Real-Time Distributed Control*, Springer, Germany, 2003.
6. H. Töpfer, W. Kriesel, Zur funktionellen und strukturellen Weiterentwicklung der Automatisierungsanlagentechnik, *Messen Steuern Regeln*, 24, 1981, 183–188.
7. T. Pfeifer, K.-U. Heiler, Ziele und Anwendungen von Feldbussystemen, *Automatisierungstechnische Praxis*, 29(12), 1987, 549–557.
8. H. Steusloff, Zielsetzungen und Lösungsansätze für eine offene Kommunikation in der Feldebene, *Automatisierungstechnik'90, VDI Berichte*, 855, 1990, 337–357.
9. L. Capetta, A. Mella, F. Russo, Intelligent field devices: User expectations, *Computing & Control Engineering Journal*, 6, December 1995, 270–272.
10. K. Wanser, Entwicklungen der Feldinstallation und ihre Beurteilung, *Automatisierungstechnische Praxis*, 27(5), 1985, 237–240.
11. J. A. H. Pfleger, Anforderungen an Feldmultiplexer, *Automatisierungstechnische Praxis*, 29, 1987, 205–209.
12. H. Junginger, H. Wehlan, Der Feldmultiplexer aus Anwendersicht, *Automatisierungstechnische Praxis*, 31, 1989, 557–564.
13. W. Schmieder, T. Tauchnitz, FuRIOS: Fieldbus and remote I/O—A system comparison, *Automatisierungstechnische Praxis*, 44, 2002, 61–70.
14. P. Pleinevaux, J.-D. Decotignie, Time critical communication networks: Field buses, *IEEE Network*, 2, 1988, 55–63.
15. E. H. Higham, Casting a crystal ball on the future of process instrumentation and process measurements, *IEEE Instrumentation and Measurement Technology Conference* (*IMTC '92*), New York, May 12–14, 1992, pp. 687–691.
16. J. P. Thomesse, Fieldbuses and interoperability, *Control Engineering Practice*, 7, 1999, 81–94.
17. J.-C. Orsini, Field bus: A user approach, Cahier Technique Schneider Electric no. 197, 2000, http://www.schneider-electric.com.tr/ftp/literature/publications/ECT197.pdf
18. R. D. Quick, S. L. Harper, HP-IL: A low-cost digital interface for portable applications, *Hewlett-Packard Journal*, 34, January 1983, 3–10.
19. Philips Semiconductor, *The I^2C-Bus Specification*, 2000, http://www.semiconductors.philips.com/buses/i2c/
20. H. Zimmermann, OSI reference model: The ISO model of architecture for open system interconnection. *IEEE Transactions on Communications*, 28(4), 1980, 425–432.
21. J. Day, H. Zimmermann, The OSI reference model, *Proceedings of the IEEE*, 71(12), 1983, 1334–1340.
22. D. J. Damsker, Assessment of industrial data network standards, *IEEE Transactions on Energy Conversion*, 3, 1988, 199–204.
23. H. A. Schutz. The role of MAP in factory integration, *IEEE Transactions on Industrial Electronics*, 35(1), 1988, 6–12.
24. B. Armitage, G. Dunlop, D. Hutchison, S. Yu, Fieldbus: An emerging communications standard, *Microprocessors and Microsystems*, 12, 1988, 555–562.
25. S. G. Shanmugham, T. G. Beaumariage, C. A. Roberts, D. A. Rollier, Manufacturing Communication: The MMS approach, *Computers and Industrial Engineering*, 28(1), 1995, 1–21.

26. T. Phinney, P. Brett, D. McGovan, Y. Kumeda, FieldBus—Real-time comes to OSI, *International Phoenix Conference on Computers and Communications*, March 27–30, 1991, Scottsdale, AZ, pp. 594–599.
27. K. Bender, Offene Kommunikation—Nutzen, Chancen, Perspektiven für die industrielle Kommunikation, *iNet'92*, 1992, Sindelfingen, Germany, 15–37.
28. T. Sauter, M. Felser, The importance of being competent—The role of competence centres in the fieldbus world, *FeT'99 Fieldbus Technology*, Springer, Magdeburg, September 1999, pp. 299–306.
29. Gesmer Updegrove LLP, Government Issues and Policy, http://www.consortiuminfo.org/government/
30. P. Leviti, IEC 61158: An offence to technicians? *IFAC International Conference on Fieldbus Systems and Their Applications, FeT 2001*, November 15–16, 2001, Nancy, France, pp. 36.
31. M. Felser, T. Sauter, The fieldbus war: History or short break between battles? in *Proceedings of the IEEE International Workshop on Factory Communication Systems*, Västerås, Sweden, 2002, pp. 73–80.
32. T. Phinney, Mopping up from bus wars, *World Bus Journal*, ISA, 2002, 22–23.
33. Z. Wang, *Internet QoS*: *Architectures and Mechanisms for Quality of Service*, Morgan Kaufmann Publishers, San Francisco, CA, 2001.
34. M. Knizak, M. Kunes, M. Manninger, T. Sauter, Modular agent design for fieldbus management, in *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation*, Barcelona, Spain, 1999, pp. 857–864.
35. T. Bangemann, Management of distributed automation systems based on Web technologies, in *Proceedings of the IFAC International Conference on Fieldbus Systems and Their Applications*, Nancy, France, 2001, pp. 254–259.
36. WordFIP (1998), *WWW & TCP/IP: A Web guide* [Online]. Available at: http://www.worldfip.org
37. S. Pérez, J. Vila, Building distributed embedded systems with RTLinux-GPL, in *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation*, Lisbon, Portugal, 2003, vol. 1, 2003, pp. 161–168.
38. C. Eckel, G. Gaderer, T. Sauter, Implementation requirements for Web-enabled appliances—A case study, in *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation*, Lisbon, Portugal, 2003, vol. 2, pp. 636–642.
39. A. Flammini, P. Ferrari, E. Sisinni, D. Marioli, A. Taroni, Sensor integration in industrial environment: From fieldbus to web sensors, *Computer Standards & Interfaces*, 25(2), 2003, 183–194.
40. A. Flammini, P. Ferrari, E. Sisinni, D. Marioli, A. Taroni, Sensor interfaces: From field-bus to Ethernet and Internet, *Sensors and Actuators A*, 101, 2002, 194–202.
41. N. Pereira, F. Pacheco, L. M. Pinho, A. Prayati, E. Nikoloutsos, A. Kalogeras, E. Hintze, H. Adamczyk, L. Rauchhaupt, Integration of TCP/IP and PROFIBUS protocols, in *Proceedings of the IEEE International Workshop on Factory Communication Systems*, Västerås, Sweden, 2002, pp. 157–164.
42. E. Nikoloutsos, A. Prayiati, A. Kalogeras, V. Kapsalis, S. Koubias, G. Papadopoulos, Integrating IP traffic into fieldbus networks, in *Proceedings of the IEEE International Symposium on Industrial Electronics*, L' Aquila, Italy, 2002, pp. 67–72.
43. G. G. Wood, State of play, *IEE Review*, 46, July 2000, 26–28.
44. K.-J. Beine, Ethernet meets INTERBUS, *Interbus Inside*, 8, 1999, 1–3.
45. International Electrotechnical Commission, IEC 61158-1, Digital data communications for measurement and control—Fieldbus for use in industrial control systems, Part 1: Introduction, 2003.
46. J.-P. Thomesse, M. Leon Chavez, Main paradigms as a basis for current fieldbus concepts, *Fieldbus Technology*, Springer, 1999, Vienna, Austria, pp. 2–15.
47. C. Diedrich, Profiles for fieldbuses—Scope and description technologies, *Fieldbus Technology*, Springer, Vienna, Austria, 1999, pp. 90–97.
48. U. Döbrich, P. Noury, ESPRIT project NOAH—Introduction, *Fieldbus Technology*, Springer, Vienna, Austria, 1999, pp. 414–422.

49. R. Simon, P. Neumann, C. Diedrich, M. Riedl, Field devices-models and their realisations, *IEEE International Conference on Industrial Technology* (*ICIT'02*), 11–14 December 2002, Bangkok, Thailand, pp. 307–312.
50. A. di Stefano, L. Lo Bello, T. Bangemann, Harmonized and consistent data management in distributed automation systems: The NOAH approach, *IEEE International Symposium on Industrial Electronics*, ISIE 2000, Cholula, Mexico, 4–8 December 2000, pp. 766–771.
51. M. Knizak, M. Kunes, M. Manninger, T. Sauter, Applying Internet management standards to fieldbus systems, *WFCS'97*, Barcelona, Spain, 1–3 October 1997, pp. 309–315.
52. M. Kunes, T. Sauter, Fieldbus-Internet connectivity: The SNMP approach, *IEEE Transactions on Industrial Electronics*, 48(6), 2001, 1248–1256.
53. M. Wollschlaeger, Integration of VIGO into directory services, *6th International P-NET Conference*, Vienna, Austria, May 1999, pp. 25–26.
54. M. Wollschlaeger, Framework for Web integration of factory communication systems, *IEEE International Conference on Emerging Technologies and Factory Automation* (*ETFA*), Antibes Juan-les-Pins, 15–18 October 2001, pp. 261–265.
55. J. D. Decotignie, A perspective on Ethernet-TCP/IP as a fieldbus, *IFAC International Conference on Fieldbus Systems and Their Applications, FeT 2001*, 15–16 November 2001, Nancy, France, pp. 138–143.
56. E. Byres, Ethernet to link automation hierarchy, *InTech Magazine*, June 1999, 44–47.
57. M. Felser, Real-time Ethernet—Industry prospective, *Proceedings of the IEEE*, 93, 2005, 1118–1129.
58. V. Schiffer, The CIP family of fieldbus protocols and its newest member—Ethernet/IP, *Conference on Emerging Technologies and Factory Automation*, *ETFA 2001*, October 15–18, 2001, Antibes Juan-Les-Pins, France, pp. 377–384.
59. M. Volz, Quo Vadis layer 7? *The Industrial Ethernet Book*, 5, March 2001, p. 21.
60. K. C. Lee, S. Lee, Performance evaluation of switched Ethernet for real-time industrial communications, *Computer Standards & Interfaces*, 24, 2002, 411–423.
61. TC65/SC65C, New work item proposal, 65C/306/NP, 2003.
62. *Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Standard 1588, 2002.
63. TC65/SC65C, Meeting minutes, 65C/318/INF, 2003.
64. J. Jasperneite, J. Feld, PROFINET: An integration platform for heterogeneous industrial communication systems, in *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation* (*ETFA*), Catania, 2005, pp. 815–822.
65. J. W. Szymanski, Embedded Internet technology in process control devices, in *Proceedings of the IEEE International Workshop on Factory Communication Systems*, Porto, Portugal, 2000, pp. 301–308.
66. http://www.ethercat.org/
67. http://www.ethernet-powerlink.com/
68. Schneider Automation (2005, June 15), *Modbus Messaging on TCP/IP Implementation Guide* [Online]. Available: http://www.modbus.org/
69. J.-P. Thomesse, Fieldbus technology in industrial automation, *Proceedings of the IEEE*, 93, 2005, 1073–1101.
70. J.-D. Decotignie, Some future directions in fieldbus research and development, *Fieldbus Technology*, Springer, New York, 1999, pp. 308–312.
71. A. Willig, K. Matheus, A. Wolisz, Wireless technology in industrial networks, *Proceedings of the IEEE*, 93, 2005, 1130–1151.
72. L. Rauchhaupt, System and device architecture of a radio based fieldbus—The RFieldbus system, *IEEE Workshop on Factory Communication Systems*, Västerås, Schweden, 2002, pp. 185–192.
73. S. Soucek, T. Sauter, G. Koller. Impact of QoS parameters on Internet-based EIA-709.1 control applications, in *Proceedings of the Annual Conference of the IEEE Industrial Electronics Society*, Sevilla, Spain, 2002, pp. 3176–3181.

74. S. Soucek, T. Sauter, Quality of service concerns in IP-based control systems, *IEEE Transactions of the Industrial Electronics*, 51, 2004, 1249–1258.
75. O. Kunert, Interconnecting field-buses through ATM, in *Proceedings of the IEEE International Workshop on Factory Communication Systems*, Barcelona, Spain, 1997, pp. 223–229.
76. I. Özçelik, H. Ekiz, Design, implementation and performance analysis of the PROFIBUS/ATM Local Bridge, *Computer Standards & Interfaces*, 26, 2004, 329–342.
77. R. Piggin, An introduction to safety-related networking, *IEE Computing & Control Engineering*, 15, April/May 2004, 34–39.
78. PROFIBUS International, *Profile for Failsafe with PROFIBUS*, *DP-Profile for Safety Applications*, Version 1.2, October 2002, http://www.profibus.com
79. INTERBUS Club, *INTERBUS Safety*, White Paper, 2003.
80. http://as-i-safety.net
81. ODVA, Safety networks: Increase productivity, reduce work-related accidents and save money, Open DeviceNet Vendor Assoc., White Paper, 2003, http://www.odva.org
82. J.-P. Froidevaux, O. Nick, M. Suzan, Use of fieldbus in safety related systems, an evaluation of World-FIP according to proven-in-use concept of IEC 61508, *WorldFIP News*, http://www.worldfip.org
83. G. Leen, D. Heffernan, Expanding automotive electronic systems, *IEEE Computer*, 35, January 2002, 88–93.
84. H. Gharavi, S. P. Kumar (eds.), Special issue on sensor networks and applications, *Proceedings of the IEEE*, 91(8), 2003, 1151–1153.
85. *Tunneling of Component Network Data over IP Channels*, EIA-852 Draft, 2000.
86. D. Dietrich, T. Sauter, Evolution potentials for fieldbus systems, in *Proceedings of the 3rd IEEE International Workshop on Factory Communication Systems*, Porto, Portugal, 2000, pp. 343–350.
87. G. Pratl, M. Lobachov, T. Sauter, Highly modular gateway architecture for fieldbus/Internet connections, in *Proceedings of the IFAC International Conference on Fieldbus Systems and Their Applications*, Nancy, France, 2001, pp. 267–273.
88. I. Calvo, M. Marcos, D. Orive, I. Sarachaga, A methodology based on distributed object-oriented technologies for providing remote access to industrial plants, *Control Engineering Practice*, 14, 2006, 975–990.
89. P. Palensky, The JEVIS service platform—Distributed energy data acquisition and management, in R. Zurawski (ed.), *The Industrial Information Technology Handbook*, CRC Press, Boca Raton, FL, 2005, Chapter 111.
90. V. Kapsalis, L. Hadellis, D. Karelis, S. Koubias, A dynamic context-aware access control architecture for e-services, *Computers & Security*, 25, 2006, 507–521.
91. M. Wollschlaeger, T. Bangemann, Maintenance portals in automation networks—Requirements, structures and model for Web-based solutions, in *Proceeding of the. IEEE International Workshop on Factory Communication Systems*, Wien, Austria, 2004, pp. 193–199.
92. A. C. Weaver, Enforcing distributed data security via Web services, in *Proceedings of the IEEE International Workshop on Factory Communication Systems*, Wien, Austria, 2004, pp. 397–402.
93. T. Sauter, Linking factory floor and the Internet, in R. Zurawski (ed.), *The Industrial Communication Technology Handbook*, CRC Press, Boca Raton, FL, 2005, Chapter 24.
94. O. Cramer Nielsen, A real time, object oriented fieldbus management system, *3rd IEEE International Workshop on Factory Communication Systems*, Porto, Portugal, 2000, pp. 335–340.
95. A. Baginski, G. Covarrubias, Open control—The standard for PC-based automation technology, *IEEE International Workshop on Factory Communication Systems*, 1–3 October 1997, pp. 329–333.
96. OPC Data Access Automation Specification, Version 2.0. OPC Foundation, October 14, 1998.
97. R. Bachmann, M. S. Hoang, P. Rieger, Component-based architecture for integrating fieldbus systems into distributed control applications, *Fieldbus Technology*, Springer-Verlag, Vienna, Austria, 1999, pp. 276–283.

98. R. Simon, M. Riedl, C. Diedrich, Integration of field devices using field device tool (fdt) on the basis of electronic device descriptions (EDD), *IEEE International Symposium on Industrial Electronics*, ISIE '03 June 9–11, 2003, pp. 189–194.
99. W. H. Moss, Report on ISO TC184/SC5/WG5 open systems application frameworks based on ISO 11898, *5th International CAN Conference (iCC'98)*, San Jose, CA, 03–05.11.1998, in *Proceedings of the Industrial Automation*, pp. 07.02–07.04.
100. M. Lobashov, G. Pratl, T. Sauter, Applicability of internet protocols for fieldbus access, in *Proceedings of the IEEE International Workshop on Factory Communication Systems*, Västerås, Sweden, 2002, pp. 205–213.
101. GSD Specification for PROFIBUS-FMS (version 1.0). PNO Karlsruhe.
102. Device Description Language specification. HART Communication Foundation, Austin, 1995.
103. T. Bray, J. Paoli, C. M. Sperberg-McQueen, *Extensible Markup Language* (*XML*) *1.0*, 1998, http://www.w3.org/TR/REC-xml.
104. M. Wollschlaeger, T. Bangemann, XML based description model as a platform for Web-based maintenance, in *2nd IEEE International Conference on Industrial Informatics* (*INDIN '04*), Berlin, Germany, 2004, pp. 125–130.
105. International Electrotechnical Commission, IEC 61804-2, Function blocks (FB) for process control—Part 2: Specification of FB concept and Electronic Device Description Language (EDDL), 2003.
106. P. Neumann, C. Diedrich, R. Simon, Engineering of field devices using device descriptions, in *IFAC World Congress 2002*, Barcelona, Spain.
107. M. Wollschlaeger, C. Diedrich, J. Müller, U. Epple, Integration of fieldbus systems into on-line asset management solutions based on fieldbus profile descriptions, in *Proceedings of the IEEE International Workshop on Factory Communication Systems*, Västerås, Sweden, 2002, pp. 89–96.
108. T. Sauter, Ch. Schwaiger, Achievement of secure Internet access to fieldbus systems, *Microprocessors and Microsystems*, 26, 2002, 331–339.
109. P. Palensky, T. Sauter, Security considerations for FAN-Internet connections, *IEEE International Workshop on Factory Communication Systems*, Porto, Portugal, 6–8 September 2000, pp. 27–35.
110. T. Bangemann, J. Hähniche, P. Neumann, Integration of fieldbus systems into computer-aided network facility management, in *Proceedings of the Annual Conference of the IEEE Industrial Electronics Society*, Aachen, Germany, 1998, pp. 1835–1840.
111. L. Hadellis, S. Koubiasa, V. Makios, An integrated approach for an interoperable industrial networking architecture consisting of heterogeneous fieldbuses, *Computers in Industry*, 49, 2002, 283–298.
112. CAMAC, A Modular instrumentation system for data handling, EUR4100e, March, 1969.
113. http://www.hit.bme.hu/people/papay/edu/GPIB/tutor.htm
114. National Instruments, GPIB Tutorial, www.raunvis.hi.is/~rol/Vefur/%E9r%20Instrupedia/CGPTUTO.PDF
115. W. Büsing, Datenkommunikation in der Leittechnik, *Automatisierungstechnische Praxis*, 28, 1986, 228–237.
116. G. Färber, *Bussysteme*, 2nd edn, Oldenbourg-Verlag, München, 1987.
117. M-Bus Usergroup, *The M-Bus: A Documentation*, Version 4.8 November 11, 1997, http://www.m-bus.com/mbusdoc/default.html
118. G. Leen, D. Heffernan, A. Dunne, Digital networks in the automotive vehicle, *IEE Computer & Control Engineering Journal*, 10, December 1999, 257–266.
119. CAN-in-Automation, CAN history, http://www.can-cia.de/can/protocol/history/
120. Condor Engineering, MIL-STD-1553 Tutorial, http://www.condoreng.com/support/downloads/tutorials/MIL-STD-1553Tutorial.PDF
121. Grid Connect, The Fieldbus Comparison Chart, http://www.synergetic.com/compare.htm
122. Interbus Club, Interbus Basics, 2001, http://www.interbusclub.com/en/doku/pdf/interbus_basics_en.pdf

123. H. Kirrmann, Industrial Automation, lecture notes, EPFL, 2004, http://lamspeople.epfl.ch/kirrmann/IA_slides.htm
124. H. Wölfel, Die Entwicklung der digitalen Prozeßleittechnik—Ein Rückblick (Teil 3), *Automatisierungstechnische Praxis*, 40, 1998, S17–S24.
125. T. Sauter, D. Dietrich, W. Kastner (eds.), *EIB Installation Bus System*, Publicis MCD, Erlangen, Germany, 2001.
126. E. B. Driscoll, The history of X10, http://home.planet.nl/~lhendrix/x10_history.htm
127. K. Thramboulidis, Development of distributed industrial control applications: The CORFU framework, in *Proceedings of the IEEE International Workshop on Factory Communication Systems*, Västerås, Sweden, 2002, pp. 39–46.
128. S. Kweon, M.-G. Cho, K. G. Shin, Soft real-time communication over Ethernet with adaptive traffic smoothing, *IEEE Transactions on Parallel and Distributed Systems*, 15, 2004, 946–959.
129. L. Lo Bello, G. Kaczynski, O. Mirabella, Improving the real-time behaviour of Ethernet networks using traffic smoothing, *IEEE Transactions on Industrial Informatics*, 1, 2005, 151–161.
130. T. Skeie, S. Johannessen, Ø. Holmeide, Timeliness of real-time IP communication in switched industrial Ethernet networks, *IEEE Transactions on Industrial Informatics*, 2, 2006, 25–39.
131. Y. Song, Time constrained communication over switched Ethernet, in *Proceedings of the IFAC International Conference on Fieldbus Systems and Their Applications*, Nancy, France, 2001, pp. 152–159.
132. J. Jasperneite, P. Neumann, M. Theis, K. Watson, Deterministic real-time communication with switched Ethernet, in *Proceedings of the IEEE International Workshop on Factory Communication Systems*, Västerås, Sweden, 2002, pp. 11–18.
133. M. Cho, K. G. Shin, On soft real-time guarantees on the Ethernet, in *Proceedings of the International Conference on Real-Time and Embedded Computing Systems and Applications*, Tainan, Taiwan, 2003, pp. 59–76.
134. S. H. Pee, R. H. Yang, J. Berge, B. Sim, Foundation fieldbus high speed Ethernet (HSE) implementation, in *Proceedings of the IEEE International Symposium on Intelligent Control*, Vancouver, Canada, 2002, pp. 777–782.
135. IEC SC65C, *Real Time Ethernet: P-NET on IP*, IEC 65C/360/NP, 2004.
136. J. Jasperneite, K. Shehab, K. Weber, Enhancements to the time synchronization standard IEEE-1588 for a system of cascaded bridges, in *Proceedings of the IEEE International Workshop on Factory Communication Systems*, Wien, Austria, 2004, pp. 239–244.
137. G. Gaderer, R. Höller, T. Sauter, H. Muhr, Extending IEEE 1588 to fault tolerant synchronization, in *Proceedings of the IEEE International Workshop on Factory Communication Systems*, Wien, Austria, 2004, pp. 353–357.
138. T. Sauter, Fieldbus systems: History and evolution, in R. Zurawski (ed.), *The Industrial Communication Technology Handbook*, CRC Press, Boca Raton, FL, 2005, pp. 7.1–7.39.
139. J.-P. Thomesse, Z. Mammeri, L. Vega, Time in distributed systems cooperation and communication models, in *Proceedings of the Fifth IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems*, Cheju Island, South Korea, 1995, pp. 41–49.
140. H. Kopetz, Event triggered vs time triggered real time systems, in: A. Karshmer and J. Nehmer, eds., *Proc. Int. Workshop on Operating Systems of the 90s and Beyond*, Lecture Notes in Computer Science, vol. 563, Springer, Berlin, 1990, pp. 87–101.
141. N. C. Audsley, A. Grigg, Timing analysis of the ARINC 629 databus for real-time applications, *Microprocessors and Microsystems*, 21, 1997, 55–61.
142. S. Cavalieri, S. Monforte, A. Corsaro, G. Scapellato, Multicycle polling scheduling algorithms for fieldbus networks, *Real-Time Systems*, 25(2–3), 2003, 157–185.

21

Real-Time Ethernet for Automation Applications

21.1 Introduction **21**-1
21.2 Structure of the IEC Standardization **21**-2
21.3 Real-Time Requirements **21**-3
User-Application Requirements • Performance Indicators of the IEC Standard
21.4 Practical Realizations **21**-6
Realization of the on Top of TCP/IP Protocols • Realization of the on Top of Ethernet • Realizations of the "Modified Ethernet"
21.5 Summary: Conclusions **21**-19
References **21**-19

Max Felser
Bern University of Applied Sciences

21.1 Introduction

International fieldbus standardization has always been a difficult endeavor. After a timely start in 1985 and a few enthusiastic years of development, the quest for the one and only comprehensive international fieldbus gradually became entangled in a web of company politics and marketing interests [1]. What followed was a protracted struggle inside European Committee for Electrotechnical Standardization (CENELEC, see www.cenelec.org) and International Electrotechnical Commission (IEC, see www.iec.ch) committees that finally ended up in the complete abandonment of the original idea. Instead of a single fieldbus, a collection of established systems was standardized. In Europe, CENELEC adopted a series of multivolume standards compiled from specifications of proven fieldbus systems. On a worldwide scale, IEC defined a matrix of protocol modules, the so-called types [2], together with guidelines how to combine the various modules into actually working fieldbus specifications [3]. With the adoption of the IEC 61158 standard [2] on the memorable date of December 31, 2000, the fieldbus war seemed to be settled just in time for the new millennium.

At the same time, in the office world, we see the penetration of the networks based on Ethernet and TCP/IP. The costs of the network infrastructure in the office world are steadily going down, and it is becoming possible to connect almost anything with everything, anywhere, with the help of the Internet technology. But in the field of automation technology, dedicated fieldbuses are used. The only barrier to access devices in the field of the automation world, from the Internet over a network connection, is the fieldbuses. Therefore, the question is why is it not possible to use Ethernet also in the automation technology?

The adoption of Ethernet technology for industrial communication between controllers, and even for communication with field devices, supports direct Internet capability in the field area, for instance, remote user interfaces via Web browser. But, it would be unacceptable if the adoption of the Ethernet technology would cause loss of features required in the field area, namely:

- Time deterministic communication
- Time-synchronized actions between field devices such as drives
- Efficient and frequent exchange of very small data records

An implicit but essential requirement is that the office Ethernet communication capability is fully retained so that the entire communication software involved remains usable.

This results in the following requirements:

- Support for migration of the office Ethernet to real-time Ethernet (RTE); see below for a definition
- Use of standard components such as bridges, Ethernet controllers, and protocol stacks as far as possible

To achieve the required higher quality of data transmission with limited jitter and disturbances due to TCP/IP data traffic, it may be necessary to develop further network components. In short, the RTE is a fieldbus specification that uses Ethernet for the lower two layers.

As a matter of fact, industrial RTE devices can neither be as cheap as in the office world (limited by the scale of industrial deployment) nor can plain Ethernet be applied to control applications demanding some sort of hard real-time behavior; for details of the argument see [4]. To cope with these limitations, many research projects proposed solutions for the introduction of quality of service, modifications to packet processing in switches, or synchronization between devices.

The IEC/SC65C* committee, in addition to the maintenance of the international fieldbus and its profile, finished a standardization project and defined additional aspects of RTE. And as in the case of the fieldbus, there are several competing solutions and their proponents represented.

This chapter will give an outline of this new document and the requirements specified for the RTE standardization. All solutions in this standard able to handle real-time (RT) requirements will be presented with their key technical features.

21.2 Structure of the IEC Standardization

All industrial protocols are defined in IEC 61158 [2]. This document is structured according to the open system interface (OSI) reference model in seven parts according to Table 21.1. In parts 2–6 all networks are identified by types. So there exist 16 different types of networks in six different parts.

In the IEC 61784 standard, different sets of profiles are collected as listed in Table 21.2. In IEC 61784-1 [3] the profile sets for continuous and discrete manufacturing relative to fieldbus use in industrial control systems are defined. Inside this first profile some version based on Ethernet technology are also defined. In the second standard IEC 61784-2 [5], additional profiles for ISO/IEC 8802-3 (Ethernet) based communication networks in RT applications are defined. To identify all these profiles a classification with communication profile families (CPF) according to Table 21.3 is introduced. Every

* IEC is organized in Technical Committees (TC) and Subcommittees (SC), TC65 deals with industrial-process measurement and control and SC65C with digital communication and has the scope to prepare standards on digital data communications subsystems for industrial-process measurement and control as well as on instrumentation systems used for research, development, and testing purposes.

TABLE 21.1 Structure of IEC 61158

IEC 61158-1	Introduction
IEC 61158-2-x	PhL: Physical Layer
IEC 61158-3-x	DLL: Data Link Layer Service
IEC 61158-4-x	DLL: Data Link Layer Protocols
IEC 61158-5-x	AL: Application Layer Services
IEC 61158-6-x	AL: Application Layers Protocol
IEC 61158-7	Network Management

Note: x indicates the related CPF.

TABLE 21.2 Standards Related with Profiles

IEC 61784-1	Profile sets for continuous and discrete manufacturing relative to fieldbus use in industrial control systems
IEC 61784-2	Additional profiles for ISO/IEC 8802-3 based communication networks in RT applications
IEC 61784-3-x	Profiles for functional safe communications in industrial networks
IEC 61784-4-x	Profiles for secure communications in industrial networks
IEC 61784-5-x	Installation profiles for communication networks in industrial control systems

Note: x indicates the related CPF.

TABLE 21.3 List of CPF

CPF1	Foundation® Fieldbus
CPF2	ControlNet™
CPF3	PROFIBUS
CPF4	P-NET®
CPF5	WorldFIP®
CPF6	INTERBUS®
CPF7	SwiftNet
CPF8	CC-Link
CPF9	HART
CPF10	Vnet/IP
CPF11	TCnet
CPF12	EtherCAT
CPF13	EPL
CPF14	EPA
CPF15	Modbus
CPF16	SERCOS

CPF is free to define a set of communication profiles (CPs). The complete set of CP and the related types are listed in Table 21.4.

Additional profiles listed in Table 21.2 cover functional safe communications, secure communications, and installation profiles for communication networks. These profiles are also separated according the same CPF, but are not discussed any further in this chapter.

21.3 Real-Time Requirements

Users of an RTE network have different requirements for different applications. These requirements are defined in [5] as performance indicators. A list of performance indicators defines the requirements for a class of applications. Every performance indicator has its limits or ranges and there exists interdependence between these performance indicators. Every CP has to define which performance indicators it fulfills in what conditions.

21.3.1 User-Application Requirements

Users of an RTE network have different requirements for different applications. One possible classification structure could be based on the delivery time:

- A low-speed class, i.e., the first class, for *human control* with delivery times around 100 ms. This timing requirement is typical for the case of humans involved in the system observation (10 pictures/s can already be seen as a low-quality movie), for engineering,

TABLE 21.4 Relation between CPF, CP, and Type of Protocol

	IEC 61784		IEC 61158 Services and Protocols			
Family	Part 1	Part 2	Phy	DLL	AL	Brand Names
Family 1						**Foundation Fieldbus (FF)**
	Profile 1/1		Type 1	Type 1	Type 9	FF-H1
	Profile 1/2		8802-3	TCP/UDP/IP	Type 5	FF-HSE
	Profile 1/3		Type 1	Type 1	Type 9	FF-H2
Family 2						**CIP**
	Profile 2/1		Type 2	Type 2	Type 2	ControlNet
	Profile 2/2	Profile 2/2	8802-3	TCP/UDP/IP	Type 2	EtherNet/IP
		Profile 2/2.1	8802-3	TCP/UDP/IP	Type 2	EtherNet/IP with time synchronization
	Profile 3/3		Type 2	Type 2	Type 2	DeviceNet
Family 3						**PROFIBUS and PROFINET**
	Profile 3/1		Type 3	Type 3	Type 3	PROFIBUS DP
	Profile 3/2		Type 1	Type 3	Type 3	PROFIBUS PA
	Profile 3/3		8802-3	TCP/IP	Type 10	PROFINET CBA
		Profile 3/4	8802-3	Type 10	Type 10	PROFINET IO Class A
		Profile 3/5	8802-3	Type 10	Type 10	PROFINET IO Class B
		Profile 3/6	8802-3	Type 10	Type 10	PROFINET IO Class C
Family 4						**P-NET**
	Profile 4/1		Type 4	Type 4	Type 4	P-NET RS-485
	Profile 4/2		Type 4	Type 4	Type 4	P-NET RS-232
		Profile 4/3				P-NET on IP
Family 5						**WorldFIP**
	Profile 5/1		Type 1	Type 7	Type 7	WorldFIP (MPS, MCS)
	Profile 5/2		Type 1	Type 7	Type 7	WorldFIP (MPS, MCS, SubMMS)
	Profile 5/3		Type 1	Type 7	Type 7	WorldFIP (MPS)
Family 6						**INTERBUS**
	Profile 6/1		Type 8	Type 8	Type 8	INTERBUS
	Profile 6/2		Type 8	Type 8	Type 8	INTERBUS TCP/IP
	Profile 6/3		Type 8	Type 8	Type 8	INTERBUS Subset
		Profile 3/4			Type 8/10	Link 3/4 to 6/1
		Profile 3/5			Type 8/10	Link 4/5 to 6/1
		Profile 3/6			Type 8/10	Link 4/6 to 6/1
Family 7						**SwiftNet** (not in the standard anymore)
Family 8						**CC-Link**
	Profile 8/1		Type 18	Type 18	Type 18	CC-Link/V1
	Profile 8/2		Type 18	Type 18	Type 18	CC-Link/V2
	Profile 8/3		Type 18	Type 18	Type 18	CC-Link/LT (Bus powered—low cost)
Family 9						**HART**
	Profile 9/1		—	—	Type 20	Universal Command (HART 6)
Family 10						**Vnet/IP**
		Profile 10/1	8802-3	UDP/IP	Type 17	Vnet/IP
Family 11						**TCnet**
		Profile 11/1	8802-3	Type 11	Type 11	TCnet
Family 12						**EtherCAT**
		Profile 12/1	Type 12	Type 12	Type 12	Simple IO
		Profile 12/2	Type 12	Type 12	Type 12	Mailbox and time synchronization
Family 13						**ETHERNET Powerlink**
		Profile 13/1	8802-3	Type 13	Type 13	
Family 14						**Ethernet for Plant Automation (EPA)**
		Profile 14/1	8802-3	UDP/TCP/IP	Type 14	EPA master to bridge
		Profile 14/2	8802-3	Type 14	Type 14	EPA bridge to device
Family 15						**Modbus-RTPS**
		Profile 15/1	8802-3	TCP/IP	Type 15	Modbus TCP
		Profile 15/2	8802-3	TCP/IP	Type 15	RTPS
Family 16						**SERCOS**
	Profile 16/1		Type 16	Type 16	Type 16	SERCOS I
	Profile 16/2		Type 16	Type 16	Type 16	SERCOS II
		Profile 16/3	8802-3	Type 16	Type 16	SERCOS III

and for process monitoring. Most processes in process automation and building control fall into this class. This requirement may be fulfilled with a standard system with TCP/IP communication channel without many problems.

- In the second class, for *process control*, the requirement is a delivery time below 10 ms. This is a requirement for most tooling machine control system like programming logic controllers (PLCs) or PC based control. To reach this timing behavior, special effort has to be taken in the RTE equipment: Powerful and expensive computer resources are needed

to handle the TCP/IP protocol in RT or the protocol stack must be simplified and reduced to get these reaction times on simple and cheap resources.

- The third and most demanding class is imposed by the requirements of *motion control*: To synchronize several axes over a network, a cycle time less than 1 ms is needed with a jitter of not more than 1 μs. This can only be reached with Ethernet network with a minimal bit rate of 100 Mbps, if both protocol medium access and hardware structure are modified.

21.3.2 Performance Indicators of the IEC Standard

The following performance indicators are defined in the CPs for RTE (IEC 61784-2):

- Delivery time
- Number of RTE end-stations
- Basic network topology
- Number of switches between RTE end-stations
- Throughput RTE
- Non-RTE bandwidth
- Time synchronization accuracy
- Non-time-based synchronization accuracy
- Redundancy recovery time

Delivery time is the time needed to convey a service data unit (SDU, message payload) from one node (source) to another node (destination). The delivery time is measured at the application layer interface. The maximum delivery time shall be stated for the two cases of no transmission errors, and one lost frame with recovery.

The *number of RTE end-stations* states the maximum number of RTE end-stations supported by a CP.

The *basic network topology* supported by a CP comprises the topologies listed in Table 21.5, or as a combination of these topologies.

The *number of switches between RTE end-stations* supported by a CP defines the possible network layout and is also an important indicator.

The *throughput RTE* is the total amount of application data by octet length on one link received per second.

Non-RTE bandwidth is the percentage of bandwidth, which can be used for non-RTE communication on one link.

Time synchronization accuracy shall indicate the maximum deviation between any two nodes' clocks.

Non-time-based synchronization accuracy indicates the maximum jitter of the cyclic behavior of any two nodes, triggered by periodic events over the network for establishing cyclic behavior.

TABLE 21.5 Possible RTE Topologies

Basic Network Topology	CP
Hierarchical star	CP m/1
Ring (loop)	CP m/2
Daisy-chain	CP m/3

Note: A real topology could be any combination of the three basic topologies.

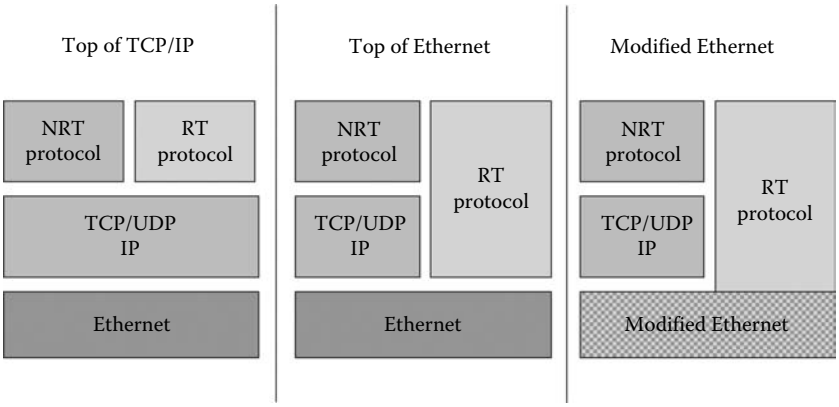


FIGURE 21.1 Possible structures for RTE.

Redundancy recovery time indicates the maximum time from a single permanent failure to the network becoming fully operational again. In this case of a permanent failure, the delivery time of a message is replaced by the redundancy recovery time.

21.4 Practical Realizations

Standard Ethernet is not able to reach the requirements of the RTE. There exist different propositions to modify the Ethernet technology by the research community [4]. The market has adopted also additional technical solutions. All the solutions included in the standardization are presented here in a short description.

Communication interfaces are structured in different layers. In Figure 21.1, a simplified structure of a communication protocol is described. Common to all Ethernet network is the universal cabling infrastructure. Non-real-time (NRT) applications make use of the Ethernet protocols as defined in ISO 8802-3 [6], and the TCP/UDP/IP protocol suite. They use typical Internet protocols like, e.g., HTTP or FTP for the NRT applications. To build an RTE solution, there are in principle three different approaches as shown in Figure 21.1. The first is to keep the TCP/UDP/IP protocols unchanged and concentrate all RT modification in the top layer; here this solution is called "on top of TCP/IP." In the second approach, the TCP/UDP/IP protocols are bypassed and the Ethernet functionality is accessed directly (on top of Ethernet); in the third approach, the Ethernet mechanism and infrastructure itself are modified to ensure RT performance (modified Ethernet).

21.4.1 Realization of the on Top of TCP/IP Protocols

Several RTE solutions use the TCP/UDP/IP protocol stack without any modification. With this protocol stack, it is possible to communicate over network boundaries transparently, also through routers. Therefore, it is possible to build automation networks reaching almost every point of the world in the same way as the Internet technology. However, the handling of this communication protocol stack requires reasonable resources in processing power and memory and introduces nondeterministic delays in the communication.

In the international standard IEC 61784-2 [5], all CPs have to list also at least one typical set of performance indicators as defined in the standard. This allows the end user an easier selection of an appropriate network for his application.

21.4.1.1 Modbus/TCP (Profiles 15/1 and 15/2)

Modbus/TCP, defined by Schneider Electric uses the well-known Modbus* over a TCP/IP network [7], using port 502 and defined as profile 15/1. This is probably one of the most widely used Ethernet solutions in industrial applications today and fulfills the requirements of the lowest class of applications which we called human control.

Modbus is a request/reply protocol (send a request frame and get back a reply frame) and offers services specified by function codes to read or write data objects. These data objects can be discrete inputs, coils,† input registers, or holding registers. In fact, this protocol is very simple and the actual definition must be extended with service definitions for the integration in international standards.

In additional to the historical Modbus protocol, new RT extensions have been defined as profile 15/2. These RT extensions use the RT publisher–subscriber (RTPS) protocol [8]. The RTPS protocol provides two main communication models: the publish–subscribe protocol, which transfers data from publishers to subscribers; and the composite state transfer protocol, which transfers state information from a writer to a reader.

In the CTS protocol, a CTS writer publishes state information as a variable which is subscribed by the CTS readers. The user data transmitted in the RTPS protocol from the publisher to one or several subscribers is called an issue. The attributes of the publication service object describe the contents (the topic), the type of the issue, and the quality (e.g., time interval) of the stream of issues that is published on the network. A subscriber defines a minimum separation time between two consecutive issues. It defines the maximum rate at which the subscription is prepared to receive issues. The persistence indicates how long the issue is valid. The strength is the precedence of the issue sent by the publication. Strength and persistence allow the receiver to arbitrate if issues are received from several matching publications. Publication relation may be best effort (as fast as possible but not faster as the minimum separation), or strict. In the case of the strict publisher–subscriber relation, the timing is ensured with a heartbeat message sent from the publisher to the subscriber (exact timing is middleware dependent) and a replied acknowledge message. The RTPS protocol is designed to run over an unreliable transport such as UDP/IP and a message is the contents (payload) of exactly one UDP/IP datagram.

In the standard, any concrete indication for values for the performance indicators is missing. They depend very strongly on the performance and implementation of the UDP/IP communication stack. So it is not possible to define an implementation independent message delivery time, for instance.

21.4.1.2 EtherNet/IP (Profiles 2/2 and 2/2.1)

EtherNet/IP,‡ defined by Rockwell and supported by the Open DeviceNet Vendor Association (ODVA, see www.odva.org) and ControlNet International (see www.controlnet.org), makes use of the common interface protocol (CIP) which is common to the following networks: EtherNet/IP, ControlNet, and DeviceNet [9].

The EtherNet/IP communication technology, standardized in IEC 61784-1 as Profile 2/2 (using type 2 specifications in IEC 61158), already provides ISO/IEC 8802-3 based RT communication. In full-duplex switched Ethernet, there is no possibility to get delays due to collisions. But in the switching device, Ethernet frames may be delayed, if an output port is busy with the

---

* Industrial de facto standard since 1979.

† In Modbus, for the representation of binary values, the term coil is used. This is originating from the ladder-logic where the coil of a relay is used to store binary information.

‡ EtherNet/IP™ is a trade name of ControlNet International, Ltd. and Open DeviceNet Vendor Association, Inc. IP stands here for Industrial Protocol.

TABLE 21.6 Performance Indicators for Ethernet/IP

Performance Indicator	Profile 2/2	Profile 2/2.1
Delivery time	130 µs to 20.4 ms	130–190 µs
Number of end-stations	2–1024	2–90
Number of switches between end-stations	1–1024	1–4
Throughput RTE	0–3.44 M octets/s	0–3.44 M octets/s
Non-RTE bandwidth	0% to 100%	0% to 100%
Time synchronization accuracy	—	≤1 µs
Non-time-based synchronization accuracy	—	—
Redundancy recovery time	—	—

transmission of an Ethernet frame. This may lead to nondeterministic delays which are not suitable for RT applications. To reduce these delays, a priority mechanism is defined in IEEE 802.3 which allows the sender of a frame to assign a priority to an Ethernet frame. A virtual bridged local area network (VLAN) tag is added into the Ethernet frame containing a VLAN-ID and a priority level 0 to 7 of the message. The Ethernet/IP RT messages get the highest priority and are transmitted by the switches before other NRT frames which results in better accuracy for the RT constraints.

In the CIPsync extensions, the clocks of the devices are synchronized with the IEEE 1588 [10] protocol (accuracy of 0.5 µs). The only problem is that delays may be introduced in the software protocol stack. Based on this time synchronization, the actions in the distributed system are executed based on the planned timing, e.g., a device sets its outputs to a defined value not based on the moment a message is received, but on the scheduled time. With this principle, the timing of the application is independent of the delay introduced in the communication network and relies only on the accuracy of the time synchronization. This is defined as profile 2/2.1. When these guidelines are strictly applied, Ethernet/IP is an RT solution usable even for the most demanding classes of applications—compare the range of values in Table 21.6—but it is still not deterministic as a communication network.

CIP defines objects* to transport control-oriented data associated with I/O devices and other information which are related to the system being controlled, such as configuration parameters and diagnostics. The CIP communication objects and application objects are grouped in classes. Profiles for different types of applications define the objects to be implemented and their relations.

21.4.1.3 P-NET (Profile 4/3)

The P-NET on IP specification has been proposed by the Danish national committee and is designed for use in an IP-environment as profile 4/3. P-NET on IP enables use of P-NET (type 4 in IEC 61158) RT communication wrapped into UDP/IP packages.

P-NET packages can be routed through IP-networks in exactly the same way as they can be routed through non-IP-networks. Routing can be through any type of P-NET network and in any order.

A P-NET frame has always two P-NET-route elements constructed as a table of destination and source addresses. In the simple case of a fieldbus solution, these two addresses are the node addresses of the fieldbus network. To allow routing over IP-based networks, these P-NET-route tables are now extended to include also IP addresses in the P-NET-route element. For a fieldbus-based P-NET node, these IP addresses are just another format of addresses. This means that any P-NET client can access servers on an IP-network without knowing anything about IP-addresses.

In fact, the P-NET on IP specification just defines how the existing P-NET package is tunneled over UDP/IP networks without any special measures to ensure RT behavior on the Ethernet network. The performance indicators are listed in Table 21.7.

* An object in CIP provides an abstract representation of a particular component within a device.

TABLE 21.7 Performance Indicators for P-NET

Performance Indicator	Profile 4/3
Delivery time	0.564–6.3 ms
Number of end-stations	30
Number of switches between end-stations	4
Throughput RTE	0–3.44 M octets/s
Non-RTE bandwidth	75%
Time synchronization accuracy	—
Non-time-based synchronization accuracy	5.7 ms
Redundancy recovery time	1 s

TABLE 21.8 Performance Indicators for Vnet/IP

Performance Indicator	Profile 10/1[a]	Profile 10/1[b]
Delivery time	20 ms	200 ms
Number of end-stations	64	4096
Number of switches between end-stations	7	39
Throughput RTE	10 M octets/s	10 M octets/s
Non-RTE bandwidth	0%–50%	0%–50%
Time synchronization accuracy	<1 ms	<5 ms
Non-time-based synchronization accuracy	—	—
Redundancy recovery time	<200 ms	<600 ms

[a] For two end-stations belonging to the same domain.
[b] For two end-stations belonging to different domains with one lost frame.

21.4.1.4 Vnet/IP (Profile 10/1)

Vnet/IP* has been developed by Yokogawa and is included in the IEC document as profile 10/1. The Vnet/IP protocol uses standard TCP/IP protocols for the integration of HTTP or other Internet protocols over the network and special RT extension protocols called RTP (RT and reliable datagram protocol).

The Vnet/IP is in fact not an RTE protocol. It just uses the UDP/IP protocol suite to transport the RTP application protocol. No special measures are taken to get a deterministic or even RT behavior. A Vnet/IP network consists of one or more domains connected to each other by routers. The IP unicast and multicast addresses are used as addresses of the data-link protocol and queued communication relations are used.

The minimum cycle time of scheduling of RT traffic is 10 ms, which fulfills the application class of process control. This specification does not cover the limiting of other traffic using the available bandwidth, e.g., HTTP or TCP transfer on the same network, which could slow down the RT behavior. The performance indicators as indicated in the IEC standard are listed in Table 21.8.

At the application layer, different objects like variables, events, regions, time and network, and the corresponding services are defined. As an example, the variable object may be accessed over client–server relations with read or write services or publisher–subscriber relations with push or pull mode of operation. In the pull model, the publisher distributes the variable data periodically by multicasting as requested by a remote subscriber. In the push model, the request is generated locally by the publisher itself.

21.4.2 Realization of the on Top of Ethernet

These RTE realizations do not alter the Ethernet communication hardware in any way, but are realized by specifying a special protocol type (Ethertype) in the Ethernet frame. The standard protocol type

* Vnet/IP is the trade name of Yokogawa Electric Corporation.

TABLE 21.9 Protocol Types for Different RTE Profiles Defined in IEC 61784

IEC 61784 Profile	Brand Name	Protocol Type
Family 3	PROFIBUS/PROFINET	0x8892
Family 11	TCnet	0x888B
Family 12	EtherCAT	0x88A4
Family 13	EPL	0x88AB
Family 14	EPA	0x88BC
Family 16	SERCOS	0x88CD

for IP is Ethertype = 0x0800. These RTE protocols do use, beside the standard IP protocol stack, their own protocol stack identified with their own protocol type. Table 21.9 lists the different values assigned to this Ethertype for these protocols.

21.4.2.1 Ethernet Powerlink (Profile 13/1)

Ethernet Powerlink (EPL) was defined by Bernecker and Rainer, and is now supported by the Ethernet Powerlink Standardisation Group (see www.ethernet-powerlink.org).

It is based on the principle of using a master–slave scheduling system on a shared Ethernet segment called Slot Communication Network Management (SCNM). The master, called managing node (MN), ensures RT access to the cyclic data and lets NRT TCP/IP frame pass through only in time slots reserved for this purpose. All other nodes are called controlled nodes (CN) and are only allowed to send on request by the MN. The MN sends a multicast Start-of-Cycle (SoC) frame to signal the beginning of a cycle. The send and receive time of this frame is the basis for the common timing of all the nodes. It is important to keep the start time of an EPL cycle as exact (jitter-free) as possible. The following time periods exist within one cycle: start period, isochronous* period, asynchronous† period, and an additional idle period. The length of individual periods can vary within the preset period of an EPL cycle. In the isochronous period of the cycle, a Poll-Request (PReq) frame is sent unicast to every configured and active node. The accessed node responds with a multicast Poll-Response (PRes) frame. In the asynchronous period of the cycle, access to the EPL network segment may be granted to one CN or to the MN for the transfer of a single asynchronous message only. The preferred protocol for asynchronous messages is UDP/IP. The Start-of-Asynchronous (SoA) frame is the first frame in the asynchronous period and is a signal for all CNs that all isochronous data has been exchanged during the isochronous period (compare also Figure 21.2). Thus the transmission of isochronous and asynchronous data will never interfere and precise communication timing is guaranteed.

An EPL network is a "protected Ethernet" defined with one controller acting as the MN and several field devices implemented as CNs. In order to protect the SCNM access mechanism of the MN, non-EPL nodes are not permitted within the protected Ethernet itself, as they would corrupt the SCNM access mechanism.

Messages exchanged between MN of different protected Ethernet segments are synchronized based on distributed clock. With the IEEE 1588 [10] protocol in every MN, a clock is synchronized and the messages between the different networks are sent based on the synchronized time in the MNs. The MN includes the routing functionality, including the IP address translation from the network to the outside world. With this synchronization mechanism, RTE communication is also possible among different networks. Performance indicators for a small- and a large-size automation system within a protected Ethernet are listed in Table 21.10.

* From Latin for iso = the same and chronous = time based, so communication at the same time interval.
† Asynchronous is without any synchronization to a reference.

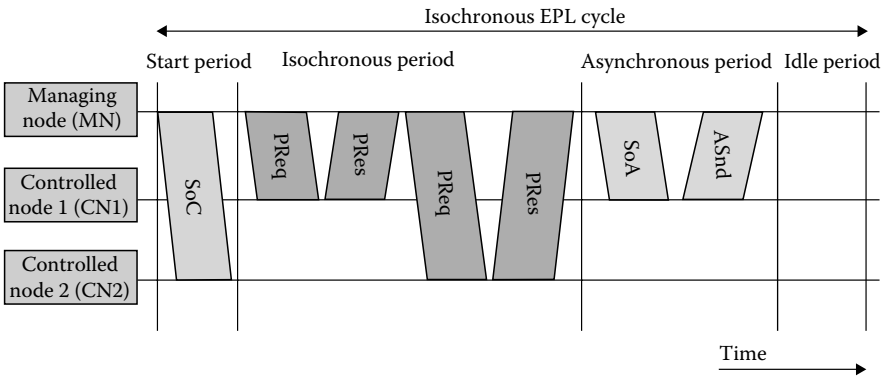


FIGURE 21.2 EPL timing.

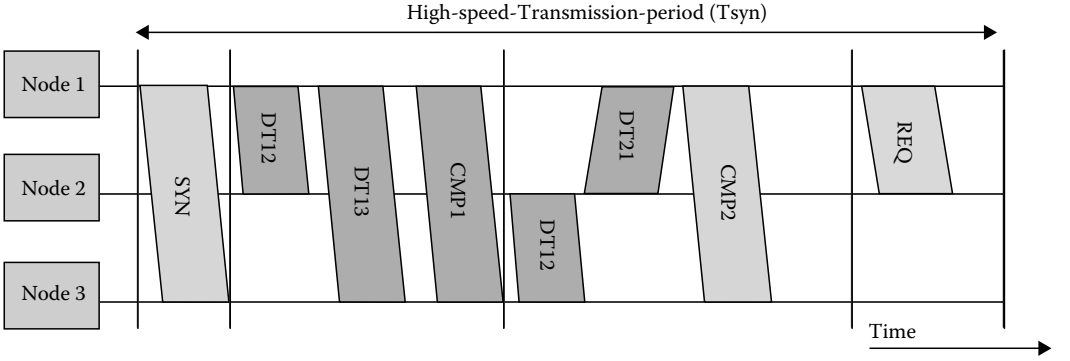


FIGURE 21.3 TCnet timing.

TABLE 21.10 Performance Indicators for EPL

Performance Indicator	Profile 13/1[a]	Profile 13/1[b]
Delivery time	400 µs	5.5 ms
Number of end-stations	4	150
Number of switches between end-stations	0 (1 Repeater)	0 (3 Repeaters)
Throughput RTE	1.9 M octets/s	4 M octets/s
Non-RTE bandwidth	19.6%	4.4%
Time synchronization accuracy	<1 s	<1 s
Non-time-based synchronization accuracy	<200 ns	<280 ns
Redundancy recovery time	150 µs	2.7 ms

[a] Small-size automation system.
[b] Large-size automation system.

The Application layer of the EPL is taken from the CANopen standards provided by the CAN in Automation (CiA, see www.can-cia.org) organization [11]. CANopen standards define widely deployed CPs, device profiles, and application profiles. Integration of EPL with CANopen combines profiles, high performance data exchange, and open, transparent communication with TCP/UDP/IP protocols. These CANopen profiles define process data objects (PDOs) to control the physical process and service data objects (SDOs) which are used to define the behavior of the device as parameters or configuration data. The PDOs are transmitted with the isochronous EPL communication, and the SDOs are transmitted with the UDP/IP protocol. Based

on this CP, a variety of CANopen device profiles can be used in an EPL environment without changes.

21.4.2.2 TCnet (Profile 11/1)

TCnet (Time-critical Control Network) is a proposal from Toshiba. Like EPL, the TCnet interface goes between the physical and the data link layer; the standard media access control (MAC) access carrier sense multiple access with collision detection (CSMA/CD) of Ethernet is modified.

In this proposal, there exists a high-speed-transmission-period composed of an RT (in TCnet called "time-critical") cyclic data service, and an asynchronous (in TCnet called "sporadic") message data service. The time-critical cyclic data service is a connection-oriented buffer transfer* on pre-established point-to-multipoint connections on the same local link separated by routers, whereas the sporadic message services are unacknowledged messages on an extended link allowed to go through routers.

At the start of the high-speed-transmission-period, a special SYN message is broadcasted to all RTE-TCnet nodes. After receiving the SYN-frame, the node with the number 1 starts sending its data frames as planned during the system configuration. After completion of the transmission of its data frames, it broadcasts a frame called Completed Message (see CMP1 in Figure 21.4). Node n upon receiving the CMP $(n-1)$ Completed Message can send out its own data frames. Each node can hold the transmission right for a preset time and must transfer the transmission right to the next node within this time. The node holding the transmission right can send cyclic data and sporadic messages. The cyclic data transmission is divided into high-, medium-, and low-speed cyclic data transmission. Each node sends at least the high-speed cyclic data when it receives the transmission right. The other, lower priority, data is send only depending on the circumstances. Thus, the cycle

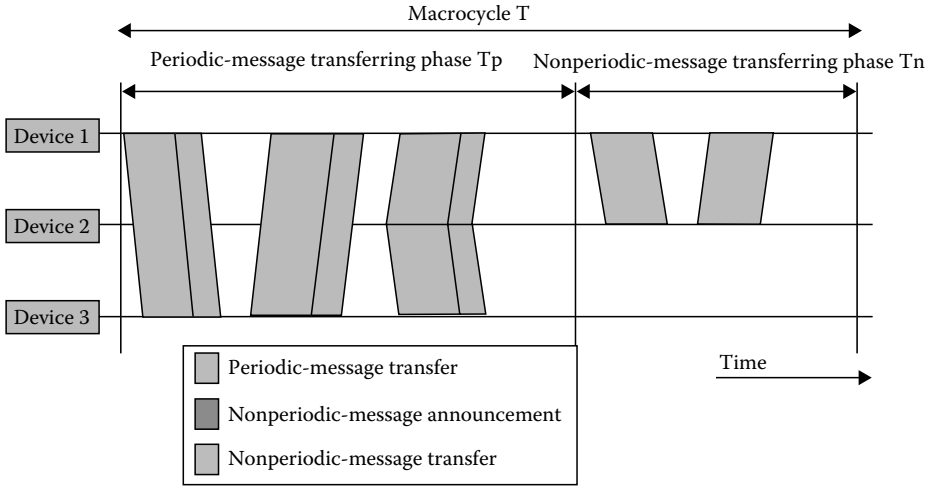


FIGURE 21.4 EPA timing.

* In a buffered transfer, a new message overwrites the old value of the previous message in the receiving buffer. This is in contrast to the (standard) queued transfer, where the messages are kept in the receiver in the same order they are send. Buffered transfer is more suited for control applications than queued; the control application is interested in the actual buffered value and not in the sequence of values.

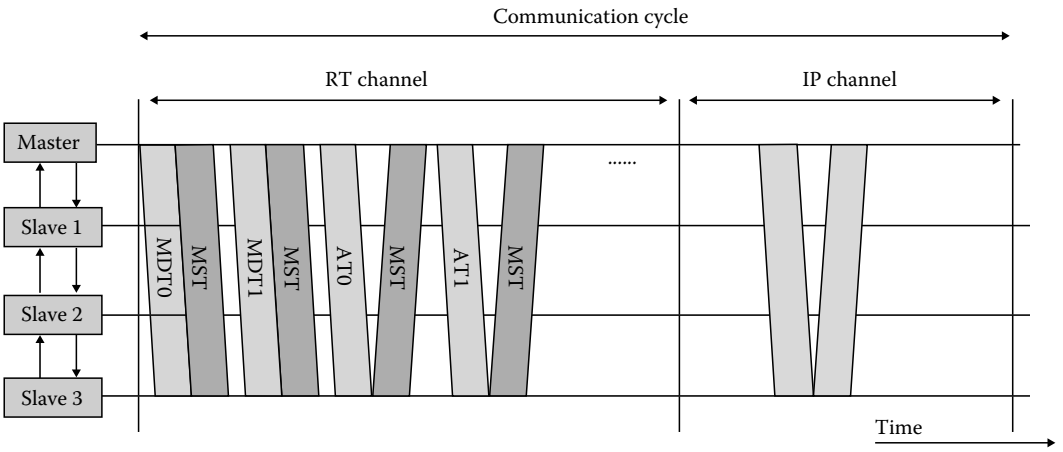


FIGURE 21.5 SERCOS timing.

time for the high-speed cycle is the cycle of the SYN frame, and the cycle time of the medium-speed or low-speed cyclic data is a multiple of the SYN frame cycle.

TCnet is able to handle redundant transmission mediums. The RTE-TCnet stack manages the selection of two redundant inputs of received frames and two outputs to two redundant transmission mediums. In the case of collision on one of the mediums, the transmission is continued on the other. The RTE-TCnet accepts the first incoming frame without transmission error from one of the redundant transmission media. This is the reason why in Table 21.11 the recovery time is set to 0.

The RTE-TCnet Application Layer Service defines the common memory system. The common memory is a virtual memory shared over the RTE-TCnet network by the participating application processes running on each node. The common memory is divided into numbers of blocks with different sizes. One node is the publisher of a block of data and broadcasts this data block to all the others by means of cyclic data service. Each node receives the data block as a subscriber and updates its local copy of the common memory. By this means, each controller can quickly access each other's data by accessing its local copy of the common memory.

21.4.2.3 EPA (Profiles 14/1 and 14/2)

The EPA* protocol (Ethernet for Plant Automation) profile 14 is a Chinese proposal.

TABLE 21.11 Performance Indicators for TCnet

Performance Indicator	Profile 11/1[a]	Profile 11/1[b]
Delivery time	2 ms/20 ms/200 ms	2 ms/20 ms/200 ms
Number of end-stations	24	13
Number of switches between end-stations	0 (3 Repeater)	0 (5 Repeaters)
Throughput RTE	7.3/6.4/0.9 M octets/s	5.7/5.1/0.6 M octets/s
Non-RTE bandwidth	0%	< 20%
Time synchronization accuracy	—	—
Non-time-based synchronization accuracy	<10 µs	<10 µs
Redundancy recovery time	0 s	0 ms

[a] With no non-RTE bandwidth.
[b] With allocated non-RTE bandwidth.

* EPA is the trade name of Zhejiang SUPCON Co. Ltd.

TABLE 21.12 Performance Indicators for EPA

Performance Indicator	Profile 14/1	Profile 14/2
Delivery time	5 ms	100 µs
Number of end-stations	32	64
Number of switches between end-stations	4	4
Throughput RTE	1.536 M octets/s	1.536 M octets/s
Non-RTE bandwidth	85%	85%
Time synchronization accuracy	<10 µs	<1 µs
Non-time-based synchronization accuracy	—	—
Redundancy recovery time	<300 ms	<300 ms

It is a distributed approach to realize deterministic communication based on a time slicing mechanism inside the MAC layer. The time to complete a communication procedure is called communication macrocycle and marked as T. Figure 21.4 illustrates that each communication macrocycle (T) is divided into two phases, periodic-message transferring phase (Tp) and nonperiodic-message transferring phase (Tn) The last part of each device's periodic message contains a nonperiodic-message announcement which indicates whether the device also has a nonperiodic message to transmit or not. Once the periodic-message transferring phase is completed, the nonperiodic-message transferring phase begins. All devices which announced (during the periodic message transfer phase) that they have a nonperiodic message to send are allowed to transmit their nonperiodic messages in this phase. Two sets of consistent performance indicators are listed in Table 21.12.

In EPA systems, there are two kinds of application processes, EPA function block* application processes and NRT application processes, which may run in parallel in one EPA system. NRT application processes are those based on regular Ethernet and TCP/IP. The interoperation between two function blocks is modeled as connecting the input/output parameters between two function blocks using EPA application services.

21.4.2.4 PROFINET CBA (Profile 3/3)

PROFINET is defined by several manufacturers (including Siemens) and supported by PROFIBUS International (see www.profibus.org) [12]. The first version was based on component-based automation (CBA) and is included in IEC 61784-1 (type 10 in IEC 61158) as profile 3/3.

The mechanical, electrical, and functional elements of an automation device are grouped together into components. Components have inputs and outputs. The values of the input and output variables of the components are transmitted over the standard TCP/IP connection using the remote procedure call (RPC)† and distributed component object model (DCOM)‡ protocol from the office world.

With this RPC and DCOM protocol it is possible to reach cycle times for what we call the human control application class. If cycle times of less than 100 ms are required, the RT protocol is used. The RT protocol is based on a special Ethertype (see Table 21.9) and frame prioritization (see explanation in Section 21.4.1.2). In this case, the TCP/IP stack is bypassed and cycle times of less than 10 ms become possible.

With PROFINET CBA, the end user defines his automation components with the traditional programming and configuration tool for PLC. These components are represented by one controller in a machine, a fieldbus network, or any device on the fieldbus itself. For the planning of the installation, logical connections between the different components are defined. These connections specify

* A function block is an algorithm with its own associated static memory. Function blocks can be instantiated with another copy of the function block's memory. Function blocks are only accessed via input and output variables.

† An RPC is a protocol that allows a computer program running on one host to cause code to be executed on another host without the programmer needing to explicitly code for this (source: wikipedia.org).

‡ DCOM is a Microsoft proprietary technology for software components distributed across several networked computers (source: wikipedia.org).

the data type and the cycle time of the transmission. The supported RT or non-RT protocols by the components define the possible cycle time which can be selected in the planning. As PROFIBUS CBA is defined in the first part of IEC61784; there are no performance indicators published.

21.4.3 Realizations of the "Modified Ethernet"

Typical cabling topology of Ethernet is the star topology, all devices are connected to a central switching device. With the introduction of the fieldbuses over 20 years ago in the automation applications, this star topology was replaced by bus or ring topologies to reduce the cabling cost. Likewise, the RTE solutions should allow for bus or ring topologies with reduced cabling effort. To permit this daisy-chained bus topology with switched Ethernet, a switch is needed in every connected device.

Most solutions providing hard RT services are based on modifications in the hardware of the device or the network infrastructure (switch or bridge). To allow cabling according to the bus or ring topology and to avoid the star topology, the switching functionality is integrated inside the field device. The modifications are mandatory for all devices inside the RT segment, but allow non-RTE traffic to be transmitted without modifications.

21.4.3.1 SERCOS (Profile 16/3)

The former IEC 61491 [13] standard SERCOS (SErial Real time COmmunication System Interface, see also www.sercos.org) is well known for its CNC (computer(ized) numerical(ly) control(led)) optical ring interface. This standard is now split into an application part and a communication part [14]; the communication part is integrated in to the IEC 61158/IEC 61784 set. The SERCOS standard is extended to feature an Ethernet-based solution with the name SERCOS III [15] as profile 16/3.

In a SERCOS system, there is always a master station as a controlling device and one or up to 254 slave devices as axis controllers each with two Ethernet ports. The basic network topology can be either a daisy-chain (line structure) or a ring (ring structure). General use switches are not permitted between any two participants. Only the free port of the last slave in a line structure may be connected to a switch if required by the configuration, e.g., for communication with devices via TCP/IP or UDP/UDP.

SERCOS III communication consists of two different logical communication channels: the RT channel (RT channel) and the IP channel (NRT channel).

The communication cycle is initiated by the master and consists of up to four master data telegrams (MDTs), and up to four device telegrams (AT*) in the RT channel and the IP channel. MDTs are transmitted by the master and received by each slave (see Figure 21.6). They contain synchronization information and a data record for each slave containing control information, service channel data, and command values sent from the master to the slaves. The ATs are transmitted by the master as an empty frame with predefined fields but without information. Each slave inserts its data into the data fields allocated to it in the ATs. Within their data fields in the telegram, the slaves transmit status information, service channel data, and actual values to the master and to other slaves.

The number and the lengths of the RT-data telegrams (MDT and AT) are fixed according to a configuration that is also determined during the initialization.

IP telegrams are standard, NRT IP telegrams that can be used for any purpose, and even be omitted. The IP channel length has a fixed duration and determines the maximum number of IP telegrams that can be sent during this duration.

* Abbreviated from device (acknowledge) telegram as AT for historical reasons.

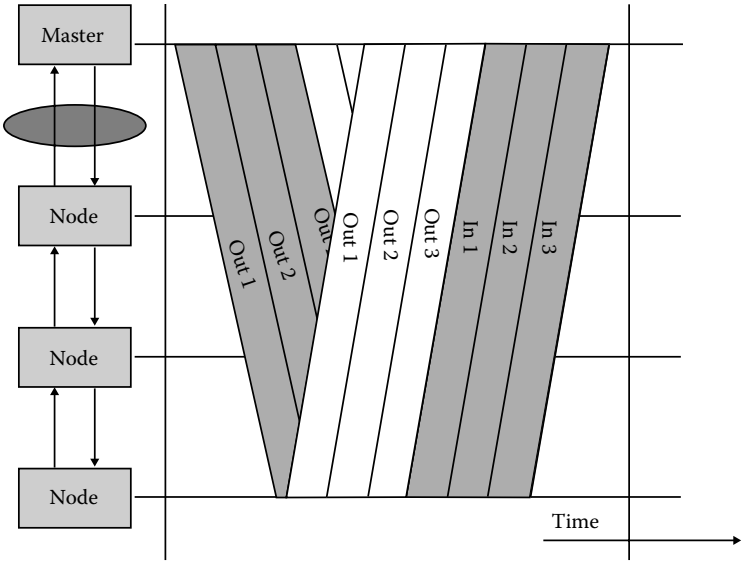


FIGURE 21.6 EtherCAT timing.

TABLE 21.13 Performance Indicators for SERCOS

Performance indicator	Profile 16/3[a]	Profile 16/3[b]
Delivery time	<39.8 µs	<513 µs
Number of end-stations	≤9	≤139
Number of switches between end-stations	0	0
Throughput RTE	11.2 M octets/s	≤9.248 M octets/s
Non-RTE bandwidth	0%	25%
Time synchronization accuracy	—	—
Non-time-based synchronization accuracy	<1 µs	<50 µs
Redundancy recovery time	0	0

[a] Scenario with minimum cycle time and high-performance synchronization.
[b] Scenario with non-RTE bandwidth and low-performance synchronization.

This sequence of transmitting synchronization, RT-data telegrams, and IP telegrams is repeated every communication cycle. Defined values a for a communication cycle are 31.25, 62.5, 125, 250 µs, and integer multiples of 250 up to 65, 000 µs. The time slots for the RT channel, the IP channel, and the transmission time of the AT are transmitted during initialization and are therefore known to each slave. In every device, a special software, or for a higher performance a field-programmable gate array (FPGA),* will be needed which separates the RT channel from the IP channel. Performance indicators for two typical setups are listed in Table 21.13.

The application model of SERCOS is based on the drive model† with a cyclic data exchange. This exchange includes status and actual values transmitted from the drive to the controller, and commands and set points from the controller to the drive. The functionality of the drive device is determined by setting different parameters in the model [14].

* FPGA, a gate array is a prefabricated circuit, with transistors and standard logic gates.
† A drive model consists of a controller and one or several drives (e.g., motors, servos).

21.4.3.2 EtherCAT (Profiles 12/1 and 12/2)

EtherCAT* defined by Beckhoff and supported by the EtherCat Technology Group (ETG, see also www.ethercat.org) uses the Ethernet frames and sends them in a special ring topology [16].

Medium access control employs the master/slave principle, where the master node (typically the control system) sends the Ethernet frames to the slave nodes, which extract data from and insert data into these frames.

From an Ethernet point of view, an EtherCAT segment is a single Ethernet device, which receives and sends standard ISO/IEC 8802-3 Ethernet frames. However, this Ethernet device is not limited to a single Ethernet controller with a downstream microprocessor, but may consist of a large number of EtherCAT slave devices. These devices process the incoming frames directly and extract the relevant user data, or insert data and transfer the frame to the next EtherCAT slave device. The last EtherCAT slave device within the segment sends the fully processed frame back, so that it is returned by the first slave device to the master as the response frame.

The EtherCAT slave node arrangement represents an open ring bus. The controller is connected to one of the open ends, either directly to the device, or via Ethernet switches utilizing the full duplex capabilities of Ethernet, the resulting topology is a physical line (see Figure 21.7). All frames are relayed from the first node to the next ones. The last node returns the telegram back to the first node, via the nodes in between.

In order to achieve maximum performance, the Ethernet frames should be processed "on the fly". This means that the node processes and relays the message to the next node in the line as the message is being received, rather than the other (slower) option of waiting until the message is fully received. If the on the fly method of processing is implemented, the slave node recognizes relevant commands and executes them accordingly while the frames are passed on to the next node. To realize such a node, a special application-specific integrated circuit (ASIC) is needed for medium access which integrates a two-port switch into the actual device.

The nodes have an addressable memory that can be accessed with read or write services, either each node consecutively or several nodes simultaneously. Several EtherCAT telegrams can be embedded within an Ethernet frame, each telegram addressing a data section.† The EtherCAT telegrams are either transported directly in the data area of the Ethernet frame or within the data section of an

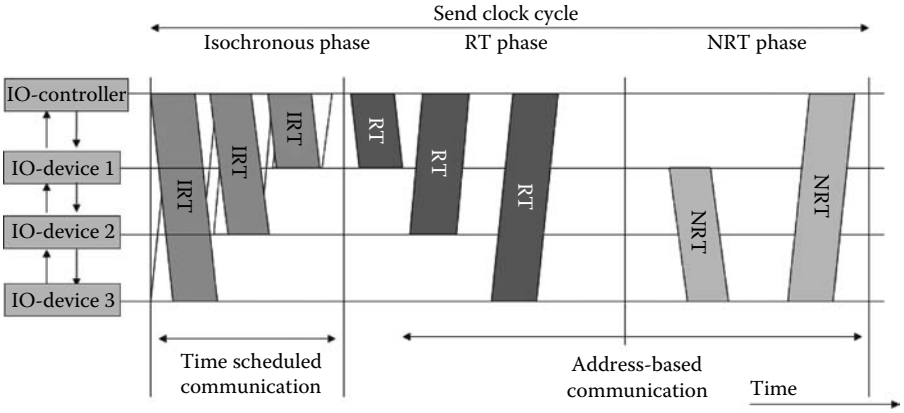


FIGURE 21.7 PROFINET timing.

* EtherCAT™ is the registered trade name of Beckhoff, Verl.

† A data section is a set of memory variables (e.g., inputs or outputs).

TABLE 21.14 Performance Indicators for EtherCAT

Performance Indicator	Profile 12/1	Profile 12/2
Delivery time	<150 µs	<519 µs
Number of end-stations	180	650
Number of switches between end-stations	NA	NA
Throughput RTE	10.75 M octets/s	10.5 M octets/s
Non-RTE bandwidth	50.6%	55.9%
Time synchronization accuracy	—	<<1 µs
Non-time-based synchronization accuracy	10 µs	10 µs
Redundancy recovery time	60 µs	200 µs

UDP datagram transported via IP. The first variant is limited to one Ethernet subnet, since associated frames are not relayed by routers. For machine control applications, this usually does not represent a constraint. Multiple EtherCAT segments can be connected to one or several switches. The Ethernet MAC address of the first node within the segment is used for addressing the EtherCAT segment. The second variant via UDP/IP generates a slightly larger overhead (IP and UDP header), but for less time-critical applications, such as building automation, it allows using IP routing. On the master side, any standard UDP/IP implementation can be used on the EtherCAT devices.

For messages, a mailbox mechanism with read and write services is used; for process data output and input, buffered data services are defined.

The performance of the EtherCAT system (when configured to run on the fly) may reach cycle times of 30 µs if no standard (non-RTE) traffic is added. The maximum transmission unit of Ethernet with 1514 bytes corresponding to approximately 125 µs at 100 MBd in the non-RTE phase would enlarge the EtherCAT cycle. Two examples of consistent sets of performance indicators are shown in Table 21.14. But in EtherCAT, Ethernet telegrams are divided into pieces and reassembled at the destination node, before being relayed as complete Ethernet telegrams to the device connected to the node (see Figure 21.6). This procedure does not restrict the achievable cycle time, since the size of the fragments can be optimized according to the available bandwidth (EtherCAT instead of IP fragmentation). This method permits any EtherCAT device to participate in the normal Ethernet traffic and still have a cycle time for RTE with less than 100 µs.

Similar to EPL, EtherCAT uses the CANopen application layer. The PDOs are mapped to the input and output buffer transfer, which is the same as what is used for EPL. The SDOs, however, are mapped to the mailbox messaging mechanism, rather than the IP protocol which EPL uses.

21.4.3.3 PROFINET IO (Profiles 3/4, 3/5, and 3/6)

PROFINET is defined by several manufacturers (including Siemens) and supported by PROFIBUS International (see www.profibus.org) [12]. A second step after the PROFINET CBA definition was the definition of an application model for PROFINET IO based on the well-proven PROFIBUS DP (type 3 of IEC 61158, profile 3/1). The devices are IO controllers to control IO devices with cyclic, buffered data communication. An IO supervisor is used to manage the IO devices and IO controllers in a system.

The exchange of data between the devices may be in different classes of communication service like isochronous RT (IRT), RT, or NRT. NRT traffic is standard TCP/UDP/IP and may also be PROFIBUS CBA traffic. In a system with high isochronous cycle requirements, only special PROFINET switching devices are allowed. The Ethernet communication is split into send clock cycles each with different time phases as presented in Figure 21.7. In the first time phase called isochronous phase, all IRT frames are transmitted. These frames are passed through the switching device without any interpretation of the address information in the Ethernet frame. The switches are set according to a predefined and configured timetable: on every offset time (see Figure 21.7), the planned frame is send from one port to the other without interpretation of the address. In the next time phase called RT phase, the switching devices change to address-based communication and behave

TABLE 21.15 Performance Indicators for PROFINET IO

Performance Indicator	Profiles 3/4 and 3/5	Profile 3/6
Delivery time	128 ms	1 ms
Number of end-stations	60	60
Number of switches between end-stations	10	20
Throughput RTE	2.324 M octets/s	3.324 M octets/s
Non-RTE bandwidth	23.5%	23.5%
Time synchronization accuracy	<1 ms	<1 μs
Non-time-based synchronization accuracy	—	—
Redundancy recovery time	<200 ms	0 ms

as standard Ethernet switches. In this addresses-based phases, RT frames are transmitted followed by NRT Ethernet frames (see also Figure 21.7). All PROFINET switching devices are synchronized by means of a modified IEEE 1588 mechanism with on the fly stamping [18], to have their cycles and IRT timetables synchronized with 1 μs jitter.

PROFINET CBA and IO do not need any special hardware for RT communication. To ensure good performance, PROFINET IO needs a 100 Mbps switched full duplex Ethernet network. For IRT, a special PROFINET–Ethernet switch is needed. It is recommended to integrate this special PROFINET–Ethernet switch in every device to allow all possible Ethernet network topologies as listed in Table 21.15.

The PROFINET specification includes a concept allowing one to integrate existing fieldbuses with proxy devices. A proxy device represents a field device or a fieldbus with several field devices, on the PROFINET network. The user of the PROFINET does not see any difference, if the device is connected to Ethernet or to the fieldbus. This proxy technology is very important to allow for a migration of the existing fieldbus installations to new Ethernet solutions with PROFINET. Initially, proxies are defined for INTERBUS (type 8 in IEC 61158) and PROFIBUS (type 3 in IEC61158) but today proxies are also defined for DeviceNet, AS-Interface, and other networks in the field.

21.5 Summary: Conclusions

During the standardization process there was a long discussion on why is it not possible to reduce the number of technical solutions to three or four RTE profiles. But there was a common understanding from the manufacturers that it is not up to the engineers in the standardization groups to take such decisions. It is agreed, that the market should decide which system will be successful in the field. So the problem of selecting a good solution is finally moved to the end user. We will see which of the RTE systems in the international standard will still be in large usage in the next 10 to 15 years.

References

1. Felser, M. and Sauter, T., The fieldbus war: History or short break between battles? *IEEE International Workshop on Factory Communication Systems* (*WFCS*), Västerås, Sweden, August 28–30, 2002, pp. 73–80.
2. IEC 61158, Digital data communications for measurement and control—Fieldbus for use in industrial control systems, 2008, available at www.iec.ch.
3. IEC 61784-1, Digital data communications for measurement and control, Part 1: Profile sets for continuous and discrete manufacturing relative to fieldbus use in industrial control systems, 2008, available at www.iec.ch.
4. Decotignie, J.-D., Ethernet-based real-time and industrial communications, *Proceedings of the IEEE*, 93(6), June 2005, 1102–1117.
5. IEC 61784-2: Industrial communication networks—Profiles, Part 2: Additional fieldbus profiles for real-time networks based on ISO/IEC 8802-3, available at www.iec.ch.

6. ISO/IEC 8802-3, Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements, Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and Physical Layer specifications, 2001.
7. Schneider Automation, Modbus messaging on TCP/IP implementation guide, May 2002, http://www.modbus.org/.
8. O. Dolejs, P. Smolik, and Z. Hanzalek, On the Ethernet use for real-time publish subscribe based applications, *2004 IEEE International Workshop on Factory Communication Systems*, September 22–24, 2004, Vienna, Austria, pp. 39–44.
9. Schiffer, V., The CIP family of fieldbus protocols and its newest member Ethernet/IP, 2001. *Proceedings of the 8th IEEE International Conference on Emerging Technologies and Factory Automation*, October 15–18, 2001, pp. 377–384, vol. 1.
10. IEEE 1588, Standard for a precision clock synchronization protocol for networked measurement and control systems, 2002.
11. CiA DS 301, CANopen application layer and communication profile, Version 4.02, February 2002.
12. PROFIBUS International, PROFINET: Technology and application, system description, Document number 4.132, April 2006, available at www.profibus.com.
13. IEC 61491, Electrical equipment of industrial machines—Serial data link for real time communication between controls and drives SERCOS, 2002–10, 2002.
14. Felser, M., Is a generic interface for power drive systems possible? *10th IEEE International Conference on Emerging Technologies and Factory Automation*, September 19–22, 2005, Catania, Italy.
15. Schemm, E., SERCOS to link with Ethernet for its third generation, *Computing & Control Engineering Journal*, 15(2), April–May 2004, 30–33.
16. Jansen, D. and Buttner, H., Real-time Ethernet the EtherCAT solution, *Computing & Control Engineering Journal*, 15(1), February–March 2004, 16–21.
17. Feld, J., PROFINET—Scalable factory communication for all applications, *2004 IEEE International Workshop on Factory Communication Systems*, September 22–24, 2004, Vienna, Austria, pp. 33–38.
18. Jasperneite, J., Shehab, K., and Weber, K., Enhancements to the time synchronization standard IEEE-1588 for a system of cascaded bridges, *2004 IEEE International Workshop on Factory Communication Systems*, September 22–24, 2004, Vienna, Austria, pp. 239–244.

22

Configuration and Management of Networked Embedded Devices

22.1 Introduction 22-1
22.2 Concepts and Terms 22-2
Configuration versus Management • Smart Devices • Plug and Play versus Plug and Participate • State
22.3 Requirements on Configuration and Management ... 22-3
22.4 Interface Separation 22-5
Interface File System Approach
22.5 Profiles, Datasheets, and Descriptions 22-7
Profiles • Electronic Device Description Language • Field Device Tool/Device Type Manager • Transducer Electronic Datasheet • Interface File System/Smart Transducer Descriptions
22.6 Application Development 22-11
22.7 Configuration Interfaces 22-14
Hardware Configuration • Plug and Participate • Application Configuration and Upload
22.8 Management Interfaces 22-16
Monitoring and Diagnosis • Calibration
22.9 Maintenance in Fieldbus Systems 22-18
22.10 Conclusion 22-19
Acknowledgment 22-19
References 22-20

Wilfried Elmenreich
University of Klagenfurt

22.1 Introduction

The advent of embedded microcontrollers and the possibility to equip them with small, fast, and low-cost network interfaces has allowed forming smart distributed systems consisting of a set of networked-embedded devices. The main reasons to build a system in a distributed way are the possibility to have redundancy and to exploit parallelism. In the context of embedded networks, a distributed system also supports the following applications:

- *Distributed sensing* with smart transducers. A smart transducer is an integration of sensor/actuator, processing element, and network interface. Smart transducers perform a local preprocessing of the analog sensor signal and transmit the measurement digitally via the network.

- Systems that integrate *legacy systems*, which cannot be extended or changed due to legal or closed system issues. Thus, the unchanged legacy system is integrated as a subsystem into a network establishing a larger overall system.
- Complexity management by dividing the overall applications into several subsystems with separate hardware and software.

Having a distributed network of embedded devices, however, also comes with increased complexity for typical configuration and management tasks such as system setup, diagnosis, repair, monitoring, calibration, change, etc. In order to handle this complexity, computer-automated mechanisms can perform tasks like registering new devices, auto-configuring data flow, detecting configuration mismatches, etc.

In this chapter, we look at the state-of-the-art mechanisms for handling these tasks. A considerable number of mature solutions for configuration and management exist in the context of fieldbus systems. A fieldbus system is a network for industrial manufacturing plants. It thus instruments fieldbus nodes with sensors, actuators, valves, console lights, switches, and contactors. Challenges for fieldbus systems are interoperability, real-time communication, robustness, and support for management and configuration. Thus, fieldbus systems have evolved a set of interesting concepts for supporting setup, configuration, monitoring, and maintenance of embedded devices connected to a fieldbus. Twenty years ago, a typical process automation plant consisted of various field devices from half a dozen vendors. Each device had its own setup program with different syntax for the same semantics. The data from the devices often differed in the data formats and the routines to interface each device [1]. Since that time, a lot of fieldbus configuration and management methods have been devised.

This chapter is organized as follows: Section 22.2 gives a definition to the concepts and terms in the context of configuration and management of networked embedded systems. Section 22.3 investigates the requirements imposed on configuration and management tasks. Section 22.4 analyzes the necessary interfaces of an intelligent device and proposes a meaningful distinction of interface types. Section 22.5 discusses profiles and other representation mechanisms of system properties for several fieldbus systems. Section 22.6 gives an overview of application development methods and their implications for configuration and management of fieldbus networks. Section 22.7 examines the initial setup of a system in terms of hardware and software configuration. Section 22.8 deals with approaches for the management of fieldbus systems, such as application download, diagnosis, and calibration of devices. Section 22.9 presents maintenance methods for reconfiguration, repair, and reintegration of networked devices. Section 22.10 concludes this overview.

22.2 Concepts and Terms

The purpose of this section is to introduce and define some important concepts and terms that are used throughout this chapter.

22.2.1 Configuration versus Management

The term *configuration* is used for a wide range of actions. Part of the configuration deals with setting up the hardware infrastructure of a fieldbus network and its nodes, i.e., physically connecting nodes (cabling) and configuring (e.g., by using switches, jumpers) nodes in a network. On the other hand, configuration also involves setting up the network on the logical (i.e., software) level. Therefore, a particular configuration mechanism often depends on the network issues like topology or underlying communication paradigm. For example, the time-triggered protocol (TTP)-Tools [2] are designed for the time-triggered architecture [3]. Thus, they provide various design and configuration mechanisms for engineering-dependable real-time systems, but it is assumed that the system is based on a time-triggered paradigm [3].

In contrast to configuration, the term *management* deals with handling of an already established system, and includes maintenance, diagnosis, monitoring, and debugging. As with configuration, different systems can greatly differ in their support and capabilities for these areas.

Often configuration and management are difficult to separate since procedures such as plug and play (see Section 22.2.3) involve configuration as well as management tasks.

22.2.2 Smart Devices

The term *smart* or *intelligent* device was first used in this context by Ko and Fung in [4], meaning a sensor or actuator device that is equipped with a network interface in order to support an easy integration into a distributed control application.

In our context, an *intelligent* device supports its configuration and management by providing its data via a well-defined network interface [5] and/or offering a self-description of its features. The description usually comes in a machine-readable form (e.g., as XML description) that resides either locally at the embedded device (e.g., IEEE1451.2 [6]) or at a higher network level being referenced by a series number (e.g., OMG [Object Management Group] smart transducer interface [7]).

22.2.3 Plug and Play versus Plug and Participate

Plug and play describes a feature for the automatic integration of a newly connected device into a system without user intervention. While this feature works well for personal computers within an office environment, it is quite difficult to achieve this behavior for automation systems, since, without user intervention, the system would not be able to guess, what sensor data should be used and what actuator should be instrumented by a given device. Therefore, in the automation domain, the more correct term *plug and participate* should be used that describes the initial configuration and integration of a new device that can be automated. For example, after connecting a new sensor to a network, it could be automatically detected, given a local name, and assigned to a communication slot. The task of a human system integrator is then reduced to decide on the further processing and usage of the sensor data.

22.2.4 State

Zadeh states that the "notion of state of a system at any given time is the information needed to determine the behavior of the system from that time on" [8, p. 3]. In real-time computer systems we distinguish between the initialization state (i-state) and the history state (h-state) [9].

The i-state encompasses the static data structure of the computer system, i.e., data that is usually located in the static (read-only) memory of the system. The i-state does not change during the execution of a given application, e.g., calibration data of a fieldbus node. The h-state is the "dynamic data structure [...] that undergoes change as the computation progresses" [9, p. 91]. Examples of an h-state are the cached results of a sequence of measurements that are used to calculate the current state of a process variable.

The size of the h-state at a given level of abstraction may vary during execution. A good system design will aim at having a *ground state*, i.e., when the size of the h-state becomes zero. In a distributed system, this usually requires that no task is active and no messages are in transit.

22.3 Requirements on Configuration and Management

The requirements imposed on a configuration and management framework are driven by several factors. We have identified the following points:

1. (*Semi*)*Automatic configuration*: The requirement for a plug-and-play-like configuration can be justified by three arguments: First, an automatic or semiautomatic configuration saves time and therefore leads to better maintainability and lower costs. Second, the necessary qualification of the person who sets up the system may be lower if the overall system is easier to configure. Third, the number of configuration faults will decrease, since monotone and error-prone tasks like looking up configuration parameters in heavy manuals are done by the computer. In most cases, a fully automatic configuration will only be possible if the functionality of the system is reduced to a manageable subset. For more complex applications, consulting a human mind is unavoidable. Thus, we distinguish two use cases, (1) the automatic setup of simple subsystems and the (2) computer-supported configuration of large distributed systems. The first case mostly deals with systems that require an automatic and autonomous (i.e., without human intervention) reconfiguration of network and communication participants in order to adapt to different operating environments. Usually, such systems either use very sophisticated (and often costly) negotiation protocols or work only on closely bounded and well-known application domains. The second case is the usual approach.
2. *Comprehensible interfaces*: In order to minimize errors, all interfaces will be made as comprehensible as possible. This includes the uniform representation of data provided by the interfaces and the capability of selectively restricting an interface to the data required by its user. The comprehensibility of an interface can be expressed by the *mental load* that it puts onto the user. Different users need different specialized interfaces, each with a minimum of mental load. For example, an application developer mostly has a service-centered view of the system. Physical network details and other properties not relevant for the application should be hidden from the developer [10].
3. *Uniform data structures*: The configuration and management of distributed embedded systems require representations of system properties that are usable by software tools. In order to avoid a situation where each application deals with the required information in its own way, these representations should be generic, highly structured, and exactly specified.
4. *Low overhead on embedded system*: Typical embedded hardware is restricted by requirements for cost, size, power consumption, and mechanical robustness. Thus, embedded hardware usually provides far less memory and processing power than average desktop systems. Currently, typical microcontrollers provide about several hundred bytes of RAM and typically less than 128 kB of Flash ROM. Clocked by an internal oscillator, these microcontrollers provide only a few million instructions per second (MIPS) of processing power. Since the local application will consume most of the available resources, the designers of configuration and management mechanisms must take care that there is not too much overhead on the embedded system nodes. This requirement drives design decisions where configuration data is compressed or even stored on a separate repository outside the embedded network.
5. *Use of standard software/hardware*: Computers running standard Windows or Linux operating systems do not provide guaranteed response times for programs, and most hardware interfaces are controlled by the operating system. Since this might violate the special timing requirements of an embedded real-time protocol, it is often not possible to directly connect a configuration host computer to the fieldbus network without a gateway. Instead, a configuration tool must use some other means of communication, such as standard communication protocols or interfaces like TCP/IP, RS232, USB, or standard middleware like CORBA (Common Object Request Broker Architecture). Often,

the system uses a dedicated gateway node which routes the configuration and management access to the sensor/actuator nodes. Thus, the timing of the embedded network is not disturbed by configuration traffic and the embedded nodes do not need to implement a USB or CORBA interface and can thus be kept slim. In order to reduce the complexity of the involved conversion and transformation steps, the interface to and from the embedded node must be comprehensible, structurally simple, and easy to access.

22.4 Interface Separation

If different user groups access a system for different purposes, they should only be provided with an interface to the information relevant for their respective purpose [11].

Interfaces for different purposes may differ by the accessible information and in the temporal behavior of the access across the interface.

Kopetz, Holzmann, and Elmenreich have identified three interfaces to a transducer node [5]:

The *configuration and planning* (CP) interface allows the integration and setup of newly connected nodes. It is used to generate the "glue" in the network that enables the components of the network to interact in the intended way. Usually, the CP interface is not time critical.

The diagnostic and management (DM) interface is used for parameterization and calibration of devices and to collect diagnostic information to support maintenance activities. For example, a remote maintenance console can request diagnostic information from a certain sensor. The DM interface is usually not time critical.

The real-time service (RS) interface is used to communicate the application data, e.g., sensor measurements or set values for an actuator. This interface usually has to fulfill timing constraints such as a bounded latency and a small communication jitter. The RS interface has to be configured by means of the CP (e.g., communication schedules) or DM interface (e.g., calibration data or level monitors).

The TTP/A fieldbus system [12] uses a time-triggered scheduling that provides a deterministic communication scheme for the RS interface. A specified part of the bandwidth is reserved for arbitrary CP and DM activities. Therefore it is possible to perform CP tasks while the system is in operation without a probe effect on the RS [13].

22.4.1 Interface File System Approach

The concept of an interface file system (IFS) was introduced in [5]. The IFS provides a unique addressing scheme to all relevant data of the nodes in a distributed system. Thus, the IFS maps real-time data, all kinds of configuration data, self-describing information, and internal state reports for diagnosis purposes.

The IFS is organized hierarchically as follows: The *cluster name* addresses a particular fieldbus network. Within the cluster, a specific node is addressed by the *node name*. The IFS of a node is structured into *files* and *records*. Each record is a unit of 4 bytes of data.

The IFS is a generic approach that has been implemented with the TTP/A protocol as a case study for the OMG smart transducer interface. The IFS approach well supports the integration and management of heterogeneous fieldbus networks. The IFS provides the following benefits:

- It establishes a well-defined interface between network communication and local application. The local application uses API (application programming interface) functions to read and write data from/into the IFS. The communication interface accesses the IFS to exchange data across the network.
- IFS provides transparency on network communication, since a task does not have to discriminate between data that is locally provided and data communicated via the network.

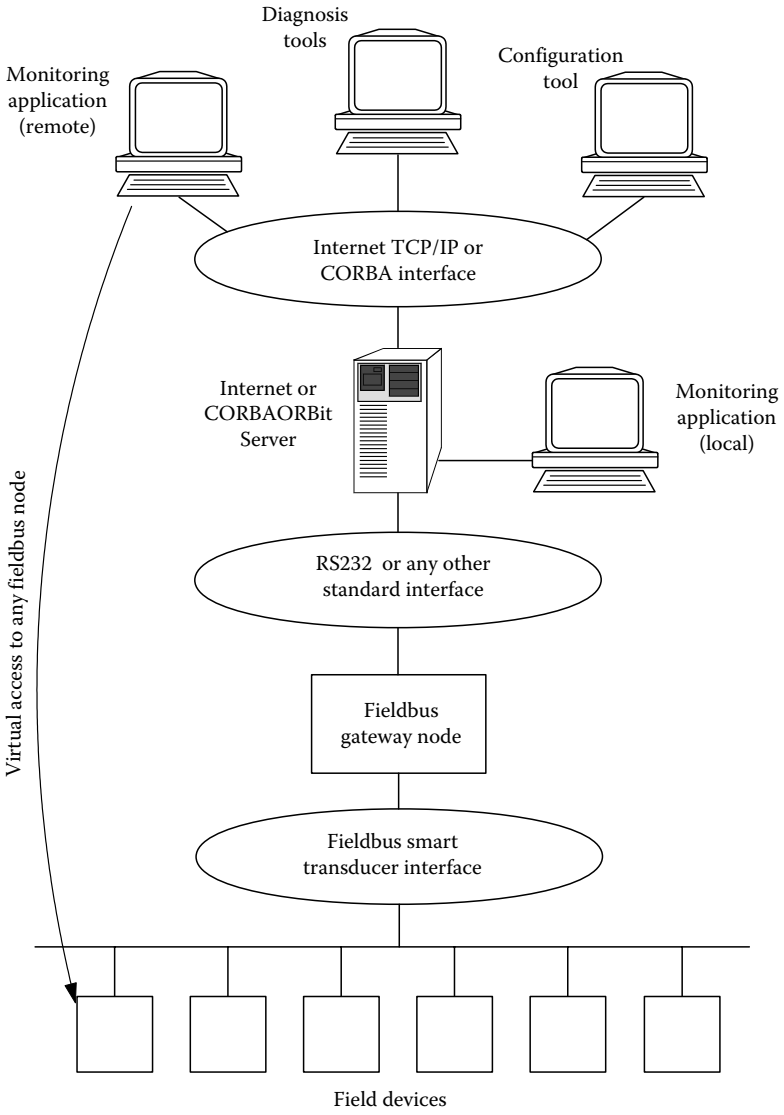


FIGURE 22.1 Architecture for remote configuration and monitoring. (From Pitzek, S. and Elmenreich, W., Configuration and management of fieldbus systems. In R. Zurawski, ed., *The Industrial Communication Technology Handbook*. CRC Press, Boca Raton, 2005. With permission.)

- Since the configuration and management data is mapped into the IFS, the IFS well supports access from configuration and management tools from outside the network. Figure 22.1 depicts an architecture with configuration and management tools that access the IFS of a fieldbus network from the Internet.

The IFS maps different data domains like RS data, configuration data, and management data using a consistent addressing scheme. In fact, the management interface can be used to define the RS data set dynamically (e.g., to select between a smoothed measurement and a measurement with better dynamics as the output from a sensor). While it is required to provide real-time guarantees for communication of real-time data, the access to configuration and management data is not time critical. This enables the employment of Web-based tools for remote maintenance.

Tools that operate on the IFS have been implemented using CORBA as middleware. CORBA is an object model managed by the OMG that provides transparent communication among remote objects. Objects can be implemented in different programming languages and can run on different platforms. The standardized CORBA protocol IIOP (Internet Inter-ORB protocol) can be routed over TCP/IP, thus supporting worldwide access to and communication between CORBA objects across the Internet.

Alternatively, it is possible to use Web Services as management interface for embedded devices. A case study that implements Web Services on top of the IFS is described in [14].

22.5 Profiles, Datasheets, and Descriptions

In order to support computer-aided configuration, dedicated computer-readable representations of network and node properties are required. This information plays a similar role as manuals and datasheets for a computer-based support framework during configuration and management of a system. These computer-readable representations allow establishing common rule sets for developing and configuring applications and for accessing devices and system properties (for configuration as well as management functions). In the following we discuss the profile approach as well as several mechanisms following an electronic datasheet approach, the Electronic Device Description Language (EDDL), the Field Device Tool/Device Type Manager (FDT/DTM), the transducer electronic datasheets (TEDS), and the smart transducer descriptions (STD) of the IFS.

22.5.1 Profiles

Profile is a widely used mechanism to create interoperability in fieldbus systems. We distinguish several types of profiles, i.e., application, functional, or device profiles. Heery and Patel propose a very general and short profile definition that we adopt for our discussion. In short, "... profiles are schemas, which consist of data elements drawn from one or more namespaces,* combined together by implementors, and optimized for a particular local application" [15].

In many cases, a profile is the result of the joint effort of a group of device vendors in a particular area of application. Usually, a task group is founded that tries to identify reoccurring functions, usage patterns, and properties in their domain and then creates strictly formalized specifications according to these identified parts, resulting in the so-called profiles.

More specific, for each device type a profile exactly defines what kind of communication objects, variables, and parameters have to be implemented so that a device conforms to the profile. Profiles usually distinguish several types of variables and parameters (e.g., process parameters, maintenance parameters, user-defined parameters) and provide a hierarchical conformance model that allows for the definition of user-defined extensions of a profile. A device profile need not necessarily correspond to a particular physical device, for example, a physical node could consist of multiple "virtual" devices (e.g., multipurpose I/O controller), or a virtual device could be distributed over several physical devices.

Protocols supporting device, functional, and application profiles are CANopen [16], PROFIBUS, and LON [17] (LonMark functional profiles). Figure 22.2 depicts, as an example, the visual specification of a LonMark† functional profile for an analog input object. The profile defines a set of network variables (in this example only the mandatory ones are shown) and local configuration parameters

* i.e., sources.

† http://www.lonmark.org.

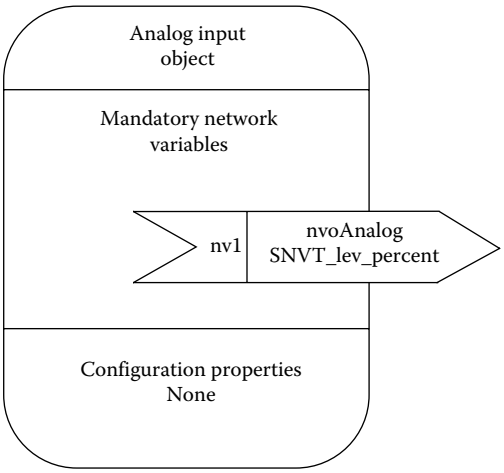


FIGURE 22.2 Functional profile for an analog input. (From Pitzek, S. and Elmenreich, W., Configuration and management of fieldbus systems. In R. Zurawski, ed., *The Industrial Communication Technology Handbook*. CRC Press, Boca Raton, 2005. With permission.)

(none in this example). The arrow specifies that this profile outputs a digital representation of an analog value, whereas the structure of this output is defined with the standardized (in LON) network variable type `SNVT_lev_percent` (–163.84% to 163.84% of full scale). In addition, a profile also specifies other important properties, such as timing information, valid range, update rate, power-up state, error condition, and behavior (usually as a state diagram).

While the approach for creating profiles is comparable in different protocols, the profiles are not always interchangeable between the various fieldbuses, although advancements (at least for process control related fieldbuses) have been made within the IEC 61158 [18] standard. Block- and class-based concepts such as function blocks, as they are defined for the Foundation Fieldbus (FF) or the PROFIBUS DP or component classes in IEEE 1451.1 [19], can be considered as implementations of the functional profile concept.

22.5.2 Electronic Device Description Language

The EDDL was first used with the HART communication protocol in 1992 [20]. In 2004, EDDL got extended by integrating the device description languages of HART, PROFIBUS, and FF, resulting in an approved international standard IEC 61804-2 [21]. An additional standard IEC 61804-3 extends EDDL by an enhanced user interface, graphical representations, and persistent data storage [22]. In 2004, the OLE for process control (OPC) Foundation joined the EDDL Cooperation Team. In succession, EDDL is used in the open OPC Unified Architecture.

In EDDL, each fieldbus component is represented by an electronic device descriptor (EDD). An EDD is represented in a text file and is operating system independent. Within the basic control and database server, an EDDL interpreter reads the EDD files corresponding to the devices present at the fieldbus system.

While EDD files are tested against various fieldbus protocols by the vendors, there is no standardized test process for assuring that an EDD works with every available EDDL interpreter. Another weak point of EDDL is that the functionality that can be described with EDDL is limited by the provided basic functions in the IEC 61804-2 standard. Thus, device functionality that cannot be described by these functions is often modeled via additional proprietary plug-ins outside the standard. With fieldbus devices becoming more and more sophisticated, it will become difficult to adequately describe devices with the current EDDL standard.

22.5.3 Field Device Tool/Device Type Manager

The FDT/DTM is a manufacturer-spanning configuration concept for fieldbus devices [23]. FDT is supported by the FDT group, which consists of currently over 50 members (vendors and users). The FTD group is responsible for the certification of DTMs.

A DTM is executable software that acts as a device driver. An FDT/DTM configuration system consists of the following parts: an FDT frame application for each distributed control system, a communication DTM (comm-DTM) for each fieldbus system, and a device DTM for each field device type. Since the device functionality is fully encapsulated into the DTM, FDT/DTM does not come with limits regarding functionality of future, eventually more complex, devices. In contrast to EDDL, FDT/DTM comes with higher costs for the device manufacturers since they have to provide the DTM device drivers. On the other hand, EDDL puts greater effort onto the system manufacturers.

The FDT software has to be executed in a separate server than the basic control and database server. Currently, DTM can only run under Microsoft Windows, which is a drawback due to Microsoft's upgrading and licensing policies (Windows versions evolve rather fast in comparison to the lifetime of an automation plant and it is not possible to get support and extra licenses for outdated Windows versions).

22.5.4 Transducer Electronic Datasheet

The TEDS was developed to establish a generic electronic datasheet format as part of the smart transducer related IEEE 1451 standards family. The IEEE 1451.2 [6] standard specifies the TEDS including the digital interface to access that datasheet and to read sensors or set actuators.

Figure 22.3 depicts the TEDS in context with the system architecture as defined in the IEEE 1451 standard:

- Smart transducer interface module (STIM): A STIM contains from 1 to 255 transducers of various predefined types together with their description in the form of the corresponding TEDS.
- Network capable application processor (NCAP): The NCAP is the interface to the overall network. By providing an appropriate NCAP, the transducer interface is independent of the physical fieldbus protocol.

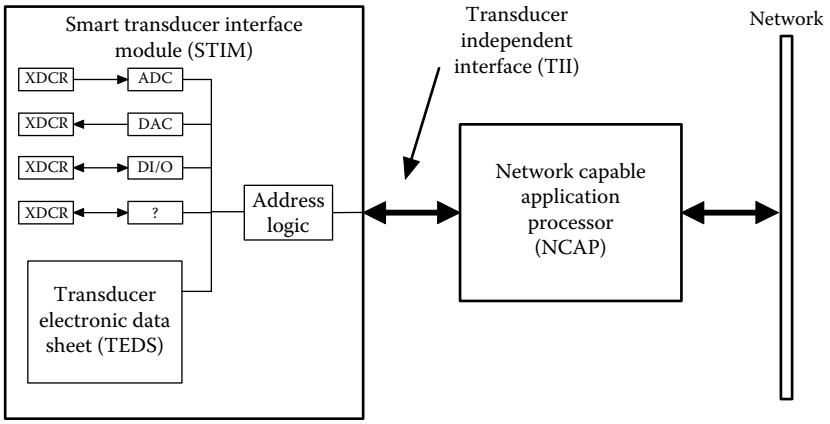


FIGURE 22.3 STIM connected to NCAP. (From Pitzek, S. and Elmenreich, W., Configuration and management of fieldbus systems. In R. Zurawski, ed., *The Industrial Communication Technology Handbook*. CRC Press, Boca Raton, 2005. With permission.)

- Transducer-independent interface (TII): The TII is the interface between the STIM and the NCAP. It is specified as an abstract model of a transducer instrumented over 10 digital communication lines.

TEDS describe node-specific properties, such as the structure and temporal properties of devices and transducer data. IEEE 1451 defines self-contained nodes with the TEDS stored in a memory directly located at the node. This causes an overhead on the nodes, so the representation of the configuration information for such a system must be compact. IEEE 1451 achieves this goal by providing a large set of predefined transducer types and nodes based on enumerated information, where identifiers are associated with more detailed prespecified descriptions (similar to error codes). Although the standard defines some possibilities to parameterize an instance of a transducer description, this approach restricts the device functionality expressiveness in a similar way as in EDDL.

22.5.5 Interface File System/Smart Transducer Descriptions

The STD as defined in [24] are a method for formally describing properties of devices that follow the CORBA STI standard (the descriptions themselves are currently not part of the standard).

The STD uses XML [25] as the primary representation mechanism for all relevant system aspects. Together with related standards, such as XML Schema or XSLT, XML provides advanced structuring, description, representation, and transformation capabilities. It is becoming the de facto standard for data representation, and has extensive support throughout the industry. Some examples where XML has already been used for applications in the fieldbus domain can be found in [26–28].

As the name implies, the STD describe the properties of nodes in the smart transducer network. The STD format is used for both describing "static" properties of a device family (comparable to classic datasheets) as well as for devices that are configured as part of a particular application (e.g., the local STD also contains the local node address). The properties described in STD cover microcontroller information (e.g., controller vendor, clock frequency, clock drift), node information (vendor name, device name/version, node identifiers), protocol information, and node service information specifying the behavior and the capabilities of a node. In the current approach, a service plays a similar role as a functional profile (see Section 22.5.1) or function block. The functional units align to the interface model of the CORBA STI standard [29].

It is not always possible to store all relevant information outside the node, but by focusing on reducing the amount of required information on the node to the minimum, extensive external meta-information can be used without size constraints. The reference to this external information is the unique combination of series and serial number of the node. The series number is identical for all nodes of the same type. The serial number identifies the instance of a node among all nodes of a series.

The advantages of this approach are twofold:

- First, the overhead at the node is very low. Current low-cost microcontrollers provide internal RAM and EEPROM memory of around 256 bytes. This will not suffice to store more than the most basic parts of datasheets according to, e.g., the IEEE 1451.2 standard without extra hardware like an external memory element. With the proposed description approach, only the memory for storing the series and serial number is necessary, which requires 8 bytes.
- Second, instead of implicitly representing the node information with many predefined data structures mapped to a compact format, it is possible to have an explicit representation of the information in a well-structured and easy-to-understand way. A typical host computer running the configuration and management tools can easily deal with even very extensive generic XML descriptions. Furthermore, XML formats are inherently easy to extend, so the format is open for future extensions of transducer or service types.

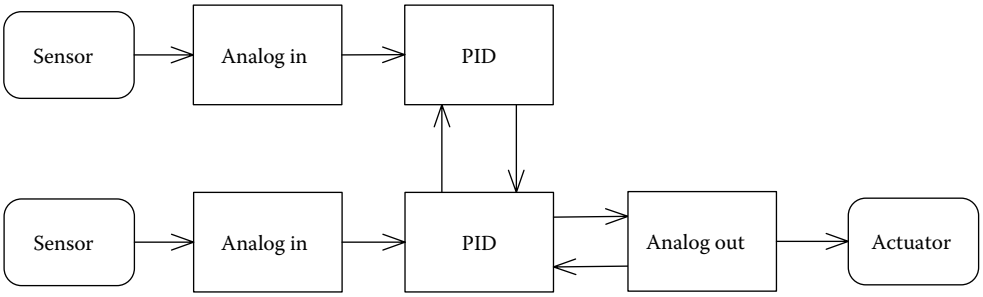


FIGURE 22.4 Example for an application model. (From Pitzek, S. and Elmenreich, W., Configuration and management of fieldbus systems. In R. Zurawski, ed., *The Industrial Communication Technology Handbook*. CRC Press, Boca Raton, 2005. With permission.)

22.6 Application Development

At the center of a fieldbus system is the actual fieldbus application. In the following we examine several application development approaches, and how they influence system configuration.

A widely used development approach for fieldbus applications is the model-based development approach. The basic idea behind this approach is to create a model of the application that consists of components that are connected via links that represent the communication flow between the components. Different approaches usually differ in what constitutes a component (e.g., function blocks, subsystems, services, functional profiles, physical devices) and the detailed semantics of a link. Many approaches support the recursive definition of components, which allows grouping multiple lower-level components into one higher-level component. Figure 22.4 depicts the model of a typical small control application consisting of two analog inputs receiving values from two sensors, two PIDs, and one analog output controlling an actuator.

But the model-based approach is not the only application design approach. Another approach used by multiple fieldbus configuration tools is the ANSI/ISA-88.01-1995 procedural control model [30]. This modeling approach enforces a strictly modular hierarchical organization of the application (see Figure 22.5). There should be no or hardly any interaction between multiple process cells. Interaction between components in a process cell is allowed. To make best use of this approach, the structure of the network site and the application should closely correspond to the hierarchy specified by this model.

The modeling approach conceptually follows the typical hierarchy of process control applications with multiple locally centralized programmable logic controllers that drive several associated control devices. This eases transition from predecessor systems and improves overall robustness, since this approach provides fault containment at the process cell level. As a downside, the coupling between the physical properties of the system and the application is rather tight. An example for a fieldbus protocol that supports this modeling approach is the PROFIBUS PA protocol that supports this model by providing a universal function block parameter for batch identification [31].

Another design approach is the two-level design approach [32], which originated in the domain of safety critical systems. In this approach the communication between components must be configured before configuring the devices. While this requires that many design decisions must be taken very early in the design process, this approach greatly improves overall composability of the components in the system.

Abstract application models provide several advantages for application development:

- Modular design of applications helps to deal with complexity by applying a "divide-and-conquer" strategy. Furthermore, it supports reuse of application components and physical separation.

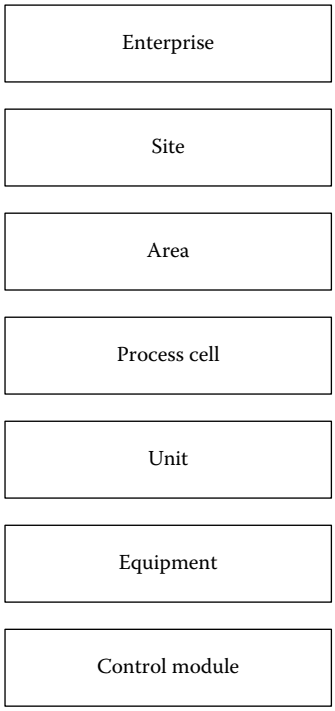


FIGURE 22.5 ANSI/ISA-88.01–1995 hierarchical model. (From Pitzek, S. and Elmenreich, W., Configuration and management of fieldbus systems. In R. Zurawski, ed., *The Industrial Communication Technology Handbook*. CRC Press, Boca Raton, 2005. With permission.)

- Separation of application logic from physical dependencies allows hardware-independent design that enables application development before the hardware is available, as well as easing migration and possibly allowing the reuse (of parts) of applications.

For configuring a physical fieldbus system from such an application model, we must examine (1) how this application model maps to the physical nodes in the network and (2) how information flow is maintained in the network.

In order to map the application model to actual devices, fieldbuses often provide a model for specifying physical devices as well. For example, in the PROFIBUS DP, the physical mapping between function blocks and the physical device is implemented as follows (see Figure 22.6). A physical device can be subdivided in several modules that take the role as virtual devices. Each device can have one (in case of simple functionality) up to many slots, which provide the mapping from the physical devices to the function blocks. A function block is mapped to a slot, whereas slots may have associated physical and transducer blocks. Physical and transducer blocks represent physical properties of a fieldbus device. Parameters of a function block are indexed, and the slot number and the parameter index cooperatively define the mapping to actual data in the device memory.

In contrast, the FF follows an object-oriented design philosophy. Thus, all information items related to configuring a device and the application (control strategy) are represented with objects. This includes function blocks, parameters, as well as subelements of parameters. These objects are collected in an object dictionary (OD), whereas each object is assigned an index. This OD defines the actual mapping to the physical memory on the respective device. In order to understand the methods for controlling the communication flow between the application components, we first examine some recurring important communication properties in fieldbus applications:

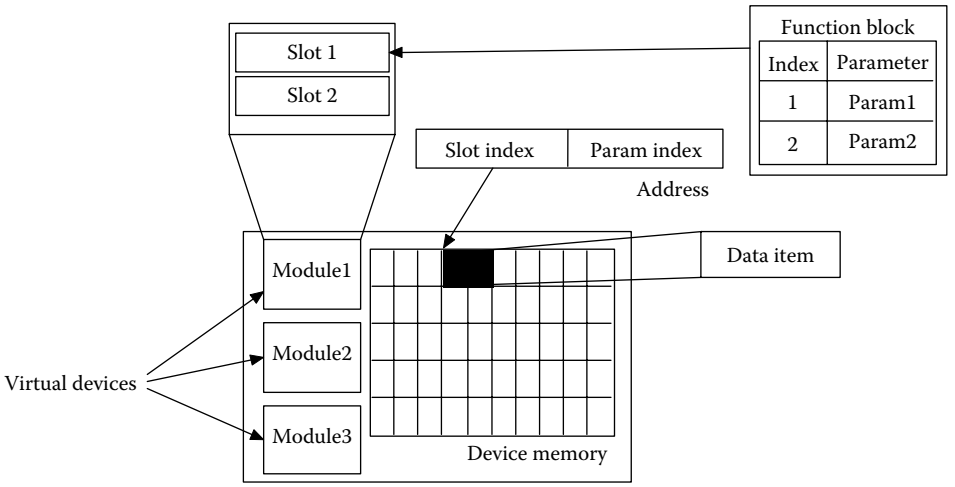


FIGURE 22.6 Mapping of function blocks to physical device in PROFIBUS DP. (From Pitzek, S. and Elmenreich, W., Configuration and management of fieldbus systems. In R. Zurawski, ed., *The Industrial Communication Technology Handbook*. CRC Press, Boca Raton, 2005. With permission.)

- Use of state communication as primary communication mechanism for operating a fieldbus [33]. State communication usually involves cyclically updating the associated application data.
- Support for asynchronous/sporadic communication (event communication) in order to perform management functions and deal with parts of the application that cannot be performed with state communication.

A common method to achieve these properties is scheduling. There are many scheduling approaches with vastly different effects on configuration. Following are some commonly used approaches adopted in fieldbus systems:

Multi-cycle polling: In this approach, communication is controlled by a dedicated node that authorizes other nodes to transmit their data [34]. This dedicated node, typically called master node, which polls the other nodes in a time division multiplexing scheme is used for bus access. This approach is taken, for example, in WorldFIP, FF, and ControlNet. For configuring the devices in such a network, the master nodes require at least a list of nodes to be polled, i.e., in case of a configuration with a single master only one node must be configured with the time information in order to control the whole cluster.

Time-triggered: In a time-triggered communication model, the communication schedule is derived from the progression of physical time. This approach requires a predefined collision-free schedule that defines *a priori* when a device is allowed to broadcast its data and an agreement on a global time, which requires the synchronization of the local clocks of all participating devices [3]. Examples for protocols that support time-triggered communication are TTP/A [12], TTP/C [35], and the synchronous part of the Flexray protocol [36]. In order to configure the communication in these systems the schedules must be downloaded to all the nodes in the network.

Event-triggered: Event-triggered communication implements a push model, where the sender decides when to send a message, e.g., when a particular value has changed more than a given *delta*. Collisions on the bus are solved by collision detection/retransmission or collision avoidance, i.e., bitwise arbitration protocols such as CAN [37]. Event-triggered communication does not depend on scheduling, since communication conflicts are either resolved by the protocol at the data link layer (e.g., bitwise arbitration) or must be resolved by the application.

The scheduling information is usually stored in dedicated data structures that are downloaded to the nodes in the network in order to be available for use by the network management system functions of the node.

The TTP/A protocol integrates the configuration information for application data flow and the communication schedule. Thus, the local communication schedules (called round descriptor lists) as well as the interfaces of application services [29] are mapped into the same interfacing mechanism, the IFS (see Section 22.4.1).

For the representation of the overall system, the cluster configuration description (CCD) format was developed that acts as a central and uniform data structure that stores all the information pertinent to the fieldbus system. This information includes

- Cluster description meta information: This description block holds information on the cluster description itself, such as the maintainer, name of the description file, or the version of the CCD format itself.
- Communication configuration information: This information includes round sequence lists as well as round descriptor lists, which represent the detailed specification of the communication behavior of the cluster. Additionally, this part of the CCD also includes (partially physical) properties important for communication, such as the universal asynchronous receiver transmitter (UART) specification, line driver, and minimum/maximum signal run-times.
- Cluster node information: This block contains information on the nodes in a cluster, whereas nodes are represented with the STD format.

22.7 Configuration Interfaces

In Section 22.6 we focused on the relation between application and configuration. In the following, we examine parts of system configuration that are mostly independent of the application. We take a short look on the physical configuration of fieldbus systems, how nodes are recognized by the configuration system, and how actual application code is downloaded to the fieldbus nodes.

22.7.1 Hardware Configuration

The hardware configuration embraces the setup of plugs and cables of the fieldbus system. Several fieldbus systems implement means to avoid mistakes, such as connecting a power cable to a sensor input, which would cause permanent damage to the fieldbus system or even harm people. Moreover, the hardware configuration interfaces such as plugs and clamps are often subject to failure in harsh environments, e.g., on a machine that induces a lot of vibration.

For hardware configuration the following approaches can be identified:

- Usage of special jacks and cables that support a tight mechanical connection and avoid mistakes in orientation and polarity by their geometry.

For example, the actuator–sensor interface* (AS-i) specifies a mechanically coded flat cable that allows the connection of slaves on any position on the cable by using piercing connectors. AS-i uses cables with two wires transporting data and energy via the same line. The piercing connectors support simple connection, safe contacting, and protection up to class IP67.

* http://www.as-interface.net/.

- Baptizing of devices in order to obtain an identifier that allows addressing the newly connected device. This could be done by explicitly assigning an identifier to the device, e.g., by setting dip switches or entering a number over a local interface, or implicitly by the cabling topology, e.g., devices could be daisy chained and obtain their name subsequently according to the chain.

Alternatively, it is possible to assign unique identifiers to nodes in advance. This approach is taken for example, with Ethernet devices where the MAC address is a worldwide unique identifier, or in the TTP/A protocol that also uses unique node IDs. However, such a worldwide unique identifier will have many digits, so that it is usually not feasible to have the number printed somewhere on the device. To overcome this problem, machine-readable identifiers in form of bar codes or RF tags are used during hardware configuration.

- Simple configuration procedures, which can be carried out and verified by nonexpert personnel.

22.7.2 Plug and Participate

Since the hardware configuration is intended to be simple, a fieldbus system should behave intelligently in order to release human personnel from error-prone tasks.

In the stage of plug and participate, the fieldbus system runs an integration task that identifies new nodes, obtains information about these nodes, and changes the network configuration in order to include the new nodes in the communication.

Identification of new nodes can be supported with manual baptizing as described in Section 22.7.1. Alternatively, it is also possible to automatically search for new nodes and identify them as described in [38].

If there can be different classes of nodes, it is necessary to obtain information on the type of the newly connected nodes. This information will usually be available in form of an electronic datasheet that can be obtained from the node or from an adequate repository.

The necessary changes of the network configuration for including the new node greatly depend on the employed communication paradigm. In case of a polling paradigm, only the list of nodes to be polled has to be extended. In case of a time-triggered paradigm, the schedule has to be changed and updated in all participating nodes. In case of an event-triggered paradigm, only the new node has to be authorized to send data; however, it is very difficult to predict how a new sender will affect the timing behavior of an event-triggered system. In all three cases critical timing might be affected due to a change of the response time, i.e., when the cycle time has to be changed. Thus, in time-critical systems, extensibility must be taken into account during system design, e.g., by reserving at first unused bandwidth or including spare communication slots.

22.7.3 Application Configuration and Upload

Some frequently reoccurring fieldbus applications, like standard feedback control loops, alert monitoring, and simple control algorithms can often be put in place like building bricks, since these applications are generically available (e.g., PID controller).

For more complex or unorthodox applications, however, it is necessary to implement user-defined applications. These cases require that code must be uploaded onto the target devices.

About 15 years ago, the most common method to reprogram a device was to have an EPROM memory chip in a socket that was physically removed from the device, erased under UV radiation, and programmed using a dedicated development system, i.e., a PC with a hardware programming device, and then put back into the system.

Today, most memory devices and microcontrollers provide an interface for in-system serial programming of flash and EEPROM memory. The hardware interface for in-system serial programming usually consists of a connector with four to six pins that is attached to either an external programming device or directly to the development PC. These programming interfaces are often proprietary to particular processor families; but there also exist some standard interfaces that support a larger variety of devices. For example, the joint test action group (JTAG) debugging interface (IEEE Standard 1149.1) also supports the upload of application code.

While the in-system serial programming approach is much more convenient than the socketed EPROM method, both approaches are conceptually quite similar, since it is still necessary to establish a separate hardware connection to the target system. A more advanced approach for uploading applications is in-programming. In this approach it is possible to program and configure a device without taking it out of the distributed target system and without using extra cables and hardware interfaces.

In-system configuration is supported by state-of-the-art flash devices, which can reprogram themselves in part by using a bootloader program. Typically, whenever a new application has to be set up, the programming system sends the node a signal causing it to enter a dedicated upload mode. During the upload phase, usually the node's service is inactive. Failures that lead to a misconfiguration must be corrected by locally connecting a programming tool.

Alternatively, application code could be downloaded via the fieldbus into the RAM memory at startup. In this case, only the bootloader resides in the persistent memory of the device and the user-defined application code has to be downloaded at startup. This approach has the advantage of being stateless, so that errors in the system are removed at the next startup. Thus, the engineers could handle many faults by a simple restart of the system. On the other hand, this approach depends on the configuration instance at startup—the system cannot be started, if the configuration instance is down. Moreover, the restart time of a system may be considerably longer.

22.8 Management Interfaces

The possibility for performing remote management operations on distributed fieldbus devices is one of the most important advantages of fieldbus systems. Wollschläger states that "in automation systems, engineering functions for administration and optimization of devices are gaining importance in comparison with control functions" [39, p. 89].

Typical management operations are monitoring, diagnosis, or node calibration. Unlike the primary fieldbus applications, which often require cyclical, multi-drop communication, these management operations usually use a one-to-one (client–server) communication style. For this reason, most fieldbus systems support both communication styles.

A central question is whether and how this management traffic influences the primary application; a problem known as probe effect [13]. System management operations that influence the timing behavior of network communication are especially critical for typical fieldbus applications (e.g., process control loops) that require exact real-time behavior.

The probe effect can be avoided by reserving a fixed amount of the bandwidth for management operations. For example, in the FF and WorldFIP protocols the application cycle (macrocycle) is chosen to be longer than strictly required by the application, and the remaining bandwidth is free for management traffic.

In order to avoid collisions within this management traffic window, adequate mechanisms for avoiding or resolving such conflicts must be used (e.g., token-passing between nodes that want to transmit management information, and priority-based arbitration).

In TTP/A, management communication is implemented by interleaving real-time data broadcasts (implemented by multipartner rounds) with the so-called master–slave rounds that open a communication channel to individual devices.

If management traffic is directly mingled with application data, such as in CAN, LonWorks, or PROFIBUS PA, care must be taken that this management traffic does not influence the primary control application. This is typically achieved by analyzing network traffic and leaving enough bandwidth headroom. For complex systems and safety-critical systems that require certain guarantees on system behavior, this analysis can become very difficult.

22.8.1 Monitoring and Diagnosis

In order to perform passive monitoring of the communication of the application, it usually suffices to listen on the bus. However, the monitoring device must have knowledge of the communication scheme used in the network, in order to be able to understand and decode the data traffic. If this scheme is controlled by the physical time, as it is the case in time-triggered networks, the monitoring node must also synchronize itself to the network.

Advanced field devices often have built-in self-diagnostic capabilities and can disclose their own status to the management system. *The handling of the control flow of diagnostic information varies for different fieldbus systems*. Typically, a diagnosis tool or the diagnosis part of the management framework will regularly check the information in the nodes. This method is called *status polling*. In some fieldbus protocols (e.g., FF), devices can also transmit status messages by themselves (alert reporting).

In general, the restrictions from the implementation of the management interface of a fieldbus protocol also apply to monitoring, since in most fieldbus systems the monitoring traffic is transmitted using the management interface.

For systems that do not provide this separation of management from application information at the protocol level, other means must be taken to ensure that monitoring does not interfere with the fieldbus application. Since status polling usually is performed periodically, it should be straightforward to reserve adequate communication resources during system design, so that the control application is not disturbed. In case of alert reporting, the central problem without adequate arbitration and scheduling mechanisms is how to avoid overloading the network in case of "alarm showers," where many devices want to send their messages at once. It can be very difficult to give timeliness guarantees (e.g., the time between an alarm occurs and the time it is received by the respective target) in such cases. The typical approach to deal with this problem (e.g., as taken in CAN) is to provide much bandwidth headroom.

For in-depth diagnosis of devices, it is sometimes also desirable to monitor operation and internals of individual field devices. This temporarily involves greater data traffic which cannot be easily reserved a priori. Therefore, the management interface must provide some flexibility on the diagnosis data in order to dynamically adjust to the proper level of detail using some kind of "Pan and Zoom" approach [40].

22.8.2 Calibration

The calibration of transducers is an important management function in many fieldbus applications. There is some ambiguity involved concerning the use of this term. Berge strictly distinguishes between calibration and range setting:

"Calibration is the correction of sensor reading and physical outputs so they match a standard" [31, p. 363]. According to this definition, calibration cannot be performed remotely since the device must be connected to a standardized reference input.

Range setting is used to move the value range of the device so that the resulting value delivers the correctly scaled percentage value. It does not require applying an input and measuring an output, thus it can be performed remotely. In the HART bus this operation is called calibration, whereas calibration is called *trim*.

Fieldbus technology does not influence the way calibration is handled, although information that is required for calibration is stored as part of the properties that describe a device. Such information could be, e.g., the minimum calibration span limit. This is the minimum distance between two calibration points within the supported operation range of a device. Additionally, calibration-related information, i.e., the individual calibration history can be stored in the devices themselves. This information is then remotely available for management tools in order to check the calibration status of devices. Together with the self-diagnosis capabilities of the field devices this allows performing a focused and proactive management strategy.

22.9 Maintenance in Fieldbus Systems

Fieldbus maintenance is the activity of keeping the system in good working order. The extensive management functions provided by fieldbus systems, such as diagnosis, and monitoring greatly help in maintaining systems. There are several different maintenance schemes that influence the way these steps are executed in detail. Choice of a particular maintenance scheme is usually motivated by the application requirements [31]:

- *Reactive maintenance* is a scheme, in which a device is only fixed after it has been found to be broken. This case should be avoided in environments where downtimes are costly (such as in factory applications). Thus, designers of such applications will usually choose more active maintenance strategies. Nonetheless, fieldbus systems also provide advantages for this scheme, since they support the fast detection of faulty devices.
- *Preventive maintenance* is a scheme, in which devices are serviced at regular intervals even if they are working correctly. This strategy prevents unexpected downtime, thus improving availability. Due to the associated costs, this approach will only be taken in safety-related applications such as in aviation, train control, or where unexpected downtimes would lead to very high costs.
- *Predictive maintenance* is similar to preventive maintenance, differing in a dynamic service interval that is optimized by using long time statistics on devices.
- *Proactive maintenance* focuses on devices that are expected to require maintenance.

Basically, maintenance involves the following steps:

- Recognizing a defective device
- Repairing (replacing) the defective device
- Reintegrating the serviced device

In fieldbus systems, faulty devices will usually be recognized via the network. This is achieved by monitoring the fieldbus nodes and the application or with devices that are capable of sending alerts (also refer to Section 22.8).

After the source of a problem has been found, the responsible node must be serviced. This often means to disconnect the node from the network. Thus, we require strategies how the system should deal with *disconnecting* the node, as well as reconnecting and *reintegrating* the replacement node.

In case the whole system must be powered down for maintenance, the faulty node can simply be replaced and the integration of the new node occurs as part of the normal initial startup process. If powering down the whole system is undesirable or even impossible (in the sense of leading to severe consequences, as in case of safety-critical applications), this process becomes more complicated. In this case, we have several options:

- *Implementation of redundancy*: This approach must be taken for safety or mission-critical devices, where operation must be continued after a device becomes defective, respectively

during replacement. A detailed presentation of redundancy and fault-tolerant systems can be found in [41].

- *Shutdown part of the application*: In case of factory communication systems that often are organized as a multilevel network and/or use a modular approach, it might be feasible to shut down a local subnetwork (e.g., a local control loop, or a process cell as defined in the ANSI/ISA-88.01-1995 standard).

The replacement node must be configured with individual node data, such as calibration data (this data usually differs between replaced and replacement node), and the *state* of a node. The state information can include

- Information that is accumulated at run-time (the history state of a system): This information must be transferred from the replaced to the replacement node.
- Timing information, so that the node can synchronize with the network: For example, in networks that use a distributed static schedule (e.g., TTP/A), each node must be configured with its part of the global schedule in order to get a network-wide consistent communication configuration.

One alternative approach for avoiding this transferring of system state is to design a stateless system in the first place. Bauer proposes a generic approach for creating stateless systems from systems with state in [42]. Another possibility is to provide well-defined reintegration points, where this state is minimized. Since fieldbus applications typically use a cyclical communication style, the start of a cycle is a "natural" reintegration point.

22.10 Conclusion

Configuration and management play an important role for distributed embedded systems. The need for configuration and management in the domain of industrial fieldbus systems has led to interesting mechanisms which evolved during the last 20 years. Most of these mechanisms can also be applied in the domain of embedded systems.

The configuration phase can be subdivided into a part that requires local interaction such as connection of hardware and setting dip switches and a part that can be done remotely via the fieldbus system. A good design requires that the local part is as simple as possible in order to simplify the interactions done by local personnel. The other part should be supported by tools that assist the system integrator in tedious and error-prone tasks such as adjusting parameters according to the datasheet of a device. Examples for systems with such a configuration support are, among others, the IEEE 1451, the FDT, and the EDDL that all employ machine-readable electronic datasheets in order to support configuration tools.

Management encompasses functions like monitoring, diagnosis, calibration, and support for maintenance. In contrast to the configuration phase, most management functions are used concurrently to the RS during operation. Some management functions, such as monitoring, may require real-time behavior for themselves. In order to avoid a probe effect on the RS, the scheduling of the fieldbus system must be designed to integrate the management traffic with the real-time traffic.

Acknowledgment

This work was supported by the Austrian FWF project TTCAR under contract No. P18060-N04.

References

1. J. Powell. The "profile" concept in fieldbus technology. Technical article, Siemens Milltronics Process Instruments Inc., 2003.
2. TTTech Computertechnik. Utilizing TTPTools in by-wire prototype projects. White Paper, 2002.
3. H. Kopetz and G. Bauer. The time-triggered architecture. *Proceedings of the IEEE*, 91(1):112–126, Jan. 2003.
4. W. H. Ko and C. D. Fung. VLSI and intelligent transducers. *Sensors and Actuators*, 2:239–250, 1982.
5. H. Kopetz, M. Holzmann, and W. Elmenreich. A universal smart transducer Interface: TTP/A. *International Journal of Computer System Science & Engineering*, 16(2):71–77, March 2001.
6. Institute of Electrical and Electronics Engineers, Inc. IEEE Std 1451.2-1997, Standard for a smart transducer interface for sensors and actuators—Transducer to micro-processor communication protocols and transducer electronic data sheet (TEDS) formats, September 1997.
7. OMG. Smart Transducers Interface V1.0. Available specification document number formal/2003-01-01, Object Management Group, Needham, MA, January 2003. Available at http://doc.omg.org/formal/2003-01-01.
8. L.A. Zadeh. The concept of system, aggregate, and state in system theory. In L.A. Zadeh and E. Polak, Editors, *System Theory*. McGraw-Hill, New York, 1969, pp. 3–42.
9. H. Kopetz, *Real-Time Systems—Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, Boston, Dordrecht, London, 1997.
10. S. Pitzek and W. Elmenreich. Managing fieldbus systems. In *Proceedings of the Work-in-Progress Session of the 14th Euromicro International Conference*, Vienna, Austria, June 2002, pp. 13–16.
11. A. Ran and J. Xu. Architecting software with interface objects. *In Proceedings of the 8th Israeli Conference on Computer-Based Systems and Software Engineering*, Ramat-Gan, Israel, June 1997, pp. 30–37.
12. H. Kopetz, et al. Specification of the TTP/A protocol. Technical report, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, 2002. Version 2.00, Available at http://www.vmars.tuwien.ac.at/ttpa/.
13. J. Gait. A probe effect in concurrent programs. *Software Practice and Experience*, 16(3):225–233, March 1986.
14. M. Venzke. Spezifikation von interoperablen Webservices mit XQuery. PhD thesis, Technische Universität Hamburg-Harburg, Hamburg-Harburg, Germany, 2003.
15. R. Heery and M. Patel. Application profiles: Mixing and matching metadata schemas. *Ariadne*, 25, September 2000. Available at http://www.ariadne.ac.uk.
16. CAN in Automation e.V. CANopen—Communication Profile for Industrial Systems, 2002. Available at http://www.can-cia.de/downloads/.
17. D. Loy, D. Dietrich, and H.-J. Schweinzer (Eds.). *Open Control Networks*. Kluwer Academic Publishing, Boston, MA, October 2001.
18. International Electrotechnical Commission (IEC). Digital data communications for measurement and control—Fieldbus for use in industrial control systems, Part 1: Overview and guidance for the IEC 61158 series, April 2003.
19. Institute of Electrical and Electronics Engineers, Inc. IEEE Std 1451.1-1999, Standard for a Smart Transducer Interface for Sensors and Actuators—Network Capable Application Processor (NCAP) Information Model, June 1999.
20. Borst Automation. Device description language. The HART book, 9, May 1999. Available at http://www.thehartbook.com/.
21. International Electrotechnical Commission (IEC). Function blocks (FB) for process control, Part 2: Specification of FB concept and Electronic Device Description Language (EDDL). IEC Standard 61804-2:2004, 2004.

22. International Electrotechnical Commission (IEC). Function blocks (FB) for process control, Part 3: Electronic Device Description Language (EDDL). IEC 61804-3:2006, 2006.
23. J. Riegert. Field Device Tool—Mastering diversity & reducing complexity, PROCESSWest, pp. 46–48, 2005.
24. S. Pitzek and W. Elmenreich. Configuration and management of a real-time smart transducer network. In *Proceedings of the 9th IEEE International Conference on Emerging Technologies and Factory Automation*, volume 1, Lisbon, Portugal, September 2003, pp. 407–414.
25. World Wide Web Consortium (W3C). *Extensible Markup Language* (*XML*) *1.0* (*Second Edition*), October 2000. Available at http://www.w3.org.
26. M. Wollschläger. A framework for fieldbus management using XML descriptions. In *Proceedings on the 2000 IEEE International Workshop on Factory Communication Systems* (*WFCS 2000*), Porto, Portugal, September 2000, pp. 3–10.
27. S. Eberle. XML-basierte Internetanbindung technischer Prozesse. In *Informatik 2000 Neue Horizonte im neuen Jahrhundert*, Springer-Verlag, Berlin Heidelberg, September 2000, pp. 356–371.
28. D. Bühler. The CANopen Markup Language—Representing fieldbus data with XML. In *Proceedings of the 26th IEEE International Conference of the IEEE Industrial Electronics Society* (*IECON 2000*), Nagoya, Japan, October 2000, pp. 2449–2454.
29. W. Elmenreich, S. Pitzek, and M. Schlager. Modeling distributed embedded applications on an interface file system. In *The 7th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, Vienna, Austria, May 2004, pp. 175–182.
30. ANSI/ISA–88.01: Batch control part 1: Models and terminology, December 1995.
31. J. Berge. Fieldbuses for process control: Engineering, operation, and maintenance. ISA–The Instrumentation, Systems, and Automation Society, 2002.
32. S. Poledna, H. Angelow, M. Glück, M. Pisecky, I. Smaili, G. Stöger, C. Tanzer, and G. Kroiss. TTP two level design approach: Tool support for composable fault-tolerant real-time systems. SAE World Congress 2000, Detroit, MI, March 2000.
33. P. Pleinevaux and J.-D. Decotignie. Time critical communication networks: Field buses. *IEEE Network*, 2(3):55–63, May 1998.
34. S. Cavalieri, S. Monforte, A. Corsaro, and G. Scapellato. Multicycle polling scheduling algorithms for fieldbus networks. *Real-Time Systems*, 25(2–3):157–185, September–October 2003.
35. TTAGroup. TTP specification version 1.1. TTAGroup, 2003. Available at http://www.ttagroup.org.
36. T. Führer, F. Hartwich, R. Hugel, and H. Weiler. Flexray—The communication system for future control systems in vehicles. SAE World Congress 2003, Detroit, MI, March 2003.
37. Robert Bosch GmbH, Stuttgart. CAN Specification Version 2.0, 1991.
38. W. Elmenreich, W. Haidinger, P. Peti, and L. Schneider. New node integration for master-slave fieldbus networks. In *Proceedings of the 20th IASTED International Conference on Applied Informatics (AI 2002)*, Innsbruck, Austria, February 2002, pp. 173–178.
39. M. Wollschläger, C. Diedrich, T. Bangemann, J. Müller, and U. Epple. Integration of fieldbus systems into on-line asset management solutions based on fieldbus profile descriptions. In *Proceedings of the 4th IEEE International Workshop on Factory Communication Systems*, Västerås, Sweden, August 2002, pp. 89–96.
40. L. Bartram, A. Ho, J. Dill, and F. Henigman. The continuous zoom: A constrained fisheye technique for viewing and navigating large information spaces. In *ACM Symposium on User Interface Software and Technology*, Pittsburg, PA, November 1995, pp. 207–215.
41. S. Poledna. *Fault-Tolerant Real-Time Systems—The Problem of Replica Determinism*. Kluwer Academic Publishers, Boston, Dordrecht, London, 1995.
42. G. Bauer. Transparent fault tolerance in a time-triggered architecture. PhD thesis, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, 2001.

23

Networked Control Systems for Manufacturing: Parameterization, Differentiation, Evaluation, and Application

23.1 Introduction **23**-1
23.2 Parameterization of Industrial Networks **23**-4
Speed and Bandwidth • Delay and Jitter • Wired and Wireless QoS Metrics • Network QoS vs. System Performance
23.3 Differentiation of Industrial Networks **23**-12
Categorization of Networks • Time-Division Multiplexing • Random Access with Collision Arbitration: CAN (CSMA/AMP) • Ethernet-Based Networks • Impact of Ethernet Application Layer Protocols: OPC • Wireless Networks
23.4 NCS Characterization **23**-24
Theoretical Perspective • Experimental Perspective • Analytical Perspective
23.5 Applications for Industrial Networks **23**-27
Networks for Control • Networks for Diagnostics • Networks for Safety • Multilevel Factory Networking Example: Reconfigurable Factory Testbed
23.6 Future Trends **23**-33
Acknowledgments **23**-34
References **23**-34

James R. Moyne
University of Michigan

Dawn M. Tilbury
University of Michigan

23.1 Introduction

Networks have become an integral part of manufacturing over the past decade, replacing point-to-point communications at all levels. At lower levels in the factory infrastructure, networks provide higher reliability, visibility, and diagnosability, and enable capabilities such as distributed control, diagnostics, safety, and device interoperability. At higher levels, networks can leverage Internet services to enable factory-wide automated scheduling, control, and diagnostics; improve data storage and visibility; and open the door to e-manufacturing.

In general, control networks can replace traditional point-to-point wired systems while providing a number of advantages. Perhaps the simplest but most important advantage is the reduced volume of wiring. Fewer physical potential points of failure, such as connectors and wire harnesses, result in

increased reliability. This advantage is further accentuated in wireless networks. Another significant advantage is that networks enable complex distributed control systems to be realized in both horizontal (e.g., peer-to-peer coordinated control among sensors and actuators) and vertical (e.g., machine to cell to system level control) directions. Other documented advantages of networks include increased capability for troubleshooting and maintenance, enhanced interchangeability and interoperability of devices, and improved reconfigurability of control systems [36].

With the return-on-investment of control networks clear, the pace of adoption continues to quicken, with the primary application being supervisory control and data acquisition (SCADA) systems [40]. These networked SCADA systems often provide a supervisory-level factory-wide solution for coordination of machine and process diagnostics, along with other factory floor and operations information. However, networks are being used at all levels of the manufacturing hierarchy, loosely defined as device, machine, cell, subsystem, system, factory, and enterprise. Within the manufacturing domain, the application of networks can be further divided into sub-domains of "control," "diagnostics," and "safety." Control network operation generally refers to communicating the necessary sensory and actuation information for closed-loop control. The control may be time-critical, such as at a computer numeric controller (CNC) or servo drive level, or event-based, such as at a programmable logic controller (PLC) level. In this sub-domain, networks must guarantee a certain level of response time determinism to be effective. Diagnostics network operation usually refers to the communication of sensory information as necessary to deduce the health of a tool, product, or system; this is differentiated from "network diagnostics," which refers to deducing the health of the network [20,28,29,61]. Systems diagnostics solutions may "close-the-loop" around the diagnostic information to implement control capabilities such as equipment shut-down or continuous process improvement; however, the performance requirements of the system are driven by the data collection, and actuation is usually event-based (i.e., not time dependent). An important quality of diagnostics networks is the ability to communicate large amounts of data, with determinism usually less important than in control networks. Issues of data compression and security can also play a large role in diagnostic networks, especially when utilized as a mechanism for communication between user and vendor to support equipment e-diagnostics [13,28,61]. Safety is the newest of the three network sub-domains, but is rapidly receiving attention in industry [39]. Here, network requirements are often driven by standards, with an emphasis on determinism (guaranteed response time), network reliability, and capability for self-diagnosis [25].

Driven by a desire to minimize cost and maximize interoperability and interchangeability, there continues to be a movement to try to consolidate around a network technology at different levels of control and across different application domains. For example, Ethernet, which was widely regarded as a high level-only communication protocol in the past, is now being utilized as a lower-level control network. This has enabled capabilities such as web-based "drill-down" (focused data access) to the sensor level [32,61]. Also, the debate continues on the consolidation of safety and control on a single network [25]. Even more recently wireless technology is being considered as a replacement for wired networks at all levels, primarily to support diagnostics, but also to support control and even safety functionality in specific instances [7,72].

This movement toward consolidation, and indeed the technical selection of networks for a particular application, revolves around evaluating and balancing Quality of Service (QoS) parameters. Multiple components (nodes) are vying for a limited network bandwidth, and they must strike a balance with factors related to the time to deliver information end-to-end between components. Two parameters that are often involved in this balance are network average speed and determinism; briefly network speed is a function of the network access time and bit transfer rate, while determinism is a measure of the ability to communicate data consistently from end-to-end within a guaranteed time. Note that this QoS issue applies to both wired and wireless network applications; however, with wireless there must be more focus on external factors that can affect the end-to-end transmission performance, reliability, and security.

Network protocols utilize different approaches to provide end-to-end data delivery. The differentiation could be at the lowest physical level (e.g., wired vs. wireless) up through the mechanism at which network access is negotiated, all the way up through application services that are supported. Protocol functionality is commonly described and differentiated utilizing the International Standards Organization–Open Systems Interconnection (ISO–OSI) seven-layer reference model [27]. The seven layers are physical, data link, network, transport, session, presentation, and application.

The network protocol, specifically the media access control (MAC) protocol component, defines the mechanism for delegating this bandwidth in such a way so that the network is "optimized" for a specific type of communication (e.g., large data packet size with low determinism vs. small data packet size with high determinism). Over the past decade "bus wars" (referring to sensor bus network technology) have resulted in serious technical debates with respect to the optimal MAC approach for different applications [18,47].

Over the past 5 years, however, it has become more and more evident that the pervasiveness of Ethernet, especially in domains outside of manufacturing control (e.g., the Internet), will result in its eventual dominance in the manufacturing control domain [9,17,53]. This movement has been facilitated in large part by the emergence of switch technology in Ethernet networks, which can increase determinism [46]. While it is not clear yet whether or not Ethernet is a candidate for safety networking, it is a strong contender in the control sub-domain, and has achieved dominance in diagnostics networking [40].

Even more recently, there has been a strong trend toward consideration of wireless as the networking medium at all levels to support all functionalities. This trend is just beginning to take shape, though, and it is not clear to what extent wireless will successfully supplant wired technologies. It is the authors' opinion that a large portion of Ethernet solutions, especially at the higher levels, and in the domains of diagnostics and, to a lesser extent, control, will be replaced by wireless over the next decade [44].

The body of research around control networks is very deep and diverse. Networks present not only challenges of timing in control systems, but also opportunities for new control directions enabled by the distribution capabilities of control networks. For example, there has been a significant amount of recent work on networked control systems (NCSs) [3,14]. Despite this rich body of work, one important aspect of control networks remains relatively untouched in the research community: the speed of the devices on the network. Practical application of control networks often reveals that device speeds dominate in determining the system performance to the point that the speed and determinism (network QoS parameters) of the network protocol are irrelevant [35,46,54]. Unfortunately, the academic focus on networks in the analysis of control network systems, often with assumptions of zero delay of devices, has served to further hide the fact that device speed is often the determining factor in assessing the NCS performance.

In light of the strong and diverse academic and industry focus on networks as well as the myriad of network technologies, the prospect of choosing a "best" network technology for a particular environment is ominous. The choice should be governed by a number of factors that balance upfront and recurring costs and performance to an objective function that best fits the particular environment. Thus, the best solution is necessarily application dependent [41]. A methodology is needed that supports the application of theory, experimental results, and analytics to a particular application domain so that the trade-offs can be quantified and a best solution determined [42].

This chapter explores the application of NCSs in the domains of control, diagnostics, and safety in manufacturing [44]. Specifically, Section 23.2 explores the parameterization of networks with respect to balancing QoS capabilities. This parameterization provides a basis for differentiating industrial network types. Section 23.3 introduces common network protocol approaches and differentiates them with respect to functional characteristics. The importance of device performance is also explored. Section 23.4 presents a method for NCS evaluation that includes theoretical, experimental, and analytical components. In Section 23.5, network applications within the domain of manufacturing are explored; these include application sub-domains of control, diagnostics and safety, as well as

different levels of control in the factory such as machine level, cell level, and system level. Within this section, an example is presented of a multilevel factory networking solution that supports networked control, diagnostics, and safety. This chapter concludes with a discussion of future trends in industrial networks with a focus on the move to wireless networking technology.

23.2 Parameterization of Industrial Networks

The function of a network is to transmit data from one node to another. Different types of industrial networks use different mechanisms for allocating the bandwidth on the network to individual nodes, and for resolving contentions among nodes. Briefly, there are three common mechanisms for allocating bandwidth: time-division multiplexing, random access (RA) with collision detection (CD), and RA with collision avoidance (CA). In time-division multiplexing, the access time to the network is allocated in a round-robin fashion among the nodes, either by passing a token (e.g., ControlNet) or having a master poll the slaves (e.g., AS-I). Because the bandwidth is carefully allocated, no collisions will occur. If RA to the network is allowed, collisions can occur if two nodes try to access the network at the same time. The collision can be destructive or nondestructive. With a destructive collision, the data are corrupted and both nodes must retransmit (e.g., Ethernet). With a nondestructive collision, one node keeps transmitting and the other backs off (e.g., CAN); in this case, the data are not corrupted. CA mechanisms (e.g., WiFi) use random delay times to minimize the probability that two nodes will try to transmit at the same time, but collisions can still occur. These mechanisms and the most common network protocols that use them will be discussed in detail in Section 23.3

Although any network protocol can be used to send data, each network protocol has its pros and cons. In addition to the protocol, the type and amount of data to be transmitted are also important when analyzing the network performance: will the network carry many small packets of data frequently, or large packets of data infrequently? Must the data arrive before a given deadline? How many nodes will be competing for the bandwidth, and how will the contention be handled?

The QoS of a network is a multidimensional parameterized measure of how well the network performs its function; the parameter measures include the speed and bandwidth of a network (how much data can be transmitted in a time interval), the delay and jitter associated with data transmission (time for a message to reach its destination, and repeatability of this time), and the reliability and security of the network infrastructure [64]. When using networks for control, it is often important to assess determinism as a QoS parameter, specifically evaluating whether message end-to-end communication times can be predicted exactly or approximately, and whether these times can be bounded.

In this section, we will review the basic QoS measures of industrial networks, with a focus on time delays, as they are typically the most important element determining the capabilities of an industrial control system. In Section 23.3, more detailed analysis of the delays for specific networks will be given. In Section 23.4, a methodology will be presented for evaluating the many dimensions of QoS along with other factors as they relate to the particular application environment.

23.2.1 Speed and Bandwidth

The "bandwidth" of an industrial network is given in terms of the number of bits that can be transmitted per second. Industrial networks vary widely in bandwidth, including CAN-based networks, which have a maximum data rate of 1 Mb/s, and Ethernet-based networks, which can support data rates upwards of 1 Gb/s,* although most networks currently used in the manufacturing industry are based on 10 or 100 Mb/s Ethernet. DeviceNet, a commonly used network in the manufacturing industry, is based on CAN and has a maximum data rate of 500 kb/s. The "speed" is the inverse of the

* 10 Gb/s solutions are available with fiber optic cabling.

data rate, thus the time to transmit 1 bit of data over the network is T_{bit} = 1 μs for 1 Mb/s CAN and 100 ns for 10 Mb/s Ethernet.

The data rate of a network must be considered together with the packet size and overhead. Data are not just sent across the network one bit at a time. Instead, data are encapsulated into packets, with headers specifying (for example) the source and destination addresses of the packet, and often a checksum for detecting the transmission errors. All industrial networks have a minimum packet size; for example the minimum packet size is 47 bits for CAN and 84 bytes for Ethernet. A minimum "interframe time" between two packets is required between subsequent messages to ensure that each packet can be distinguished individually; this time is specified by the network protocol and is included herein as part of the frame.

The transmission time for a message on the network can be computed from the network's data rate, the message size, and the distance between two nodes. As most of these quantities can be computed exactly (or approximated closely), transmission time is considered a deterministic parameter in a network system. The transmission time can be written as the sum of the frame time and the propagation time,

$$T_{tx} = T_{frame} + T_{prop}$$

where

T_{frame} is the time required to send the packet across the network
T_{prop} is the time for a message to propagate between any two devices

As the typical transmission speed in a communication medium is 2×10^8 m/s, the propagation time T_{prop} is negligible on a small scale. In the worst case, the propagation delays from one end to the other of the network cable for two typical control networks are T_{prop} = 67.2 μs for 2500 m Ethernet,* and T_{prop} = 1 μs for 100 m CAN. The propagation delay is not easily characterized because the distance between the source and destination nodes is not constant among different transmissions, but typically it is less than 1 μs (if the devices are less than 100 m apart). Some networks (e.g., Ethernet) are not a single trunk but have multiple links connected by hubs, switches, and/or routers that receive, store, and forward packets from one link to another. The delays associated with these interconnections can dominate propagation delays in a complex network, and must also be considered when determining the transmission delays [48].

The frame time, T_{frame}, depends on the size of the data, the overhead, any padding, and the bit time. Let N_{data} be the size of the data in terms of bytes, N_{ovhd} be the number of bytes used as overhead, N_{pad} be the number of bytes used to pad the remaining part of the frame to meet the minimum frame size requirement, and N_{stuff} be the number of bytes used in a stuffing mechanism (on some protocols).† The frame time can then be expressed by the following equation:

$$T_{frame} = \left[N_{data} + N_{ovhd} + N_{pad} + N_{stuff}\right] \times 8 \times T_{bit} \tag{23.1}$$

In Ref. [33], these values are explicitly described for Ethernet, ControlNet, and DeviceNet protocols.

The effective bandwidth of a control network will depend not only on the physical bandwidth, but also on the efficiency of encoding the data into packets (how much overhead is needed in terms of addressing and padding), how efficiently the network operates in terms of (long or short) interframe times, and whether network time is wasted due to message collisions. For example, to send one bit of data over a 500 kb/s CAN network, a 47 bit message is needed, requiring 94 μs. To send the same one bit of data over 10 Mb/s Ethernet, an 84 byte message is needed (64 byte frame size plus 20 bytes for interframe separation), requiring a 67.2 μs T_{frame}. Thus, even though the raw network speed is 20

* Because Ethernet uses Manchester biphase encoding, two bits are transmitted on the network for every bit of data.
† The bit-stuffing mechanism in DeviceNet is as follows: if more than 5 bits in a row are "1," then a "0" is added and vice versa. Ethernet uses Manchester biphase encoding, and, therefore, does not require bit stuffing.

times faster for Ethernet, the frame time is only 30% lower than CAN. This example shows that the network speed is only one factor that must be considered when computing the effective bandwidth of a network.

23.2.2 Delay and Jitter

The "time delay" on a network is the total time between the data being available at the source node (e.g., sampled from the environment or computed at the controller) and it being available at the destination node (received and decoded, where the decode level depends on where the delay is evaluated within the end-to-end communication). The "jitter" is the variability in the delay. Many control techniques have been developed for systems with constant time delays [11,58], but variable time delays can be much more difficult to compensate for, especially if the variability is large. Although time delay is an important factor to consider for control systems implemented over industrial networks, it has not been well-defined or studied by standards organizations defining network protocols [66].

To further explain the different components that go into the time delay and jitter on a network, consider the timing diagram in Figure 23.1 showing how messages are sent across a network. The source node A captures (or computes) the data of interest. There is some preprocessing that must be done to encapsulate the data into a message packet and encode it for sending over the network; this time is denoted T_{pre}. If the network is busy, the node may need to wait for some time T_{wait} for the network to become available. This waiting time is a function of the MAC mechanism of the protocol, which is categorized as part of layer 2 of the OSI model. Then, the message is sent across the network, taking time T_{tx} as described in Section 23.2.1. Finally, when the message is received at the destination node B, it must be decoded and postprocessed, taking time T_{post}. Thus, the total time delay can be expressed by the following equation:

$$T_{delay} = T_{pre} + T_{wait} + T_{tx} + T_{post} \tag{23.2}$$

The waiting time T_{wait} can be computed based on the network traffic, how many nodes there are, the relative priority of these nodes and the messages they are sending, and how much data they send. The pre- and postprocessing times T_{pre} and T_{post} depend on the devices. Often the network encoding and decoding are implemented in software or firmware. These times are rarely given as part of device specifications. As they can be the major sources of delay and jitter in a network, a more detailed discussion of these delays is given here.

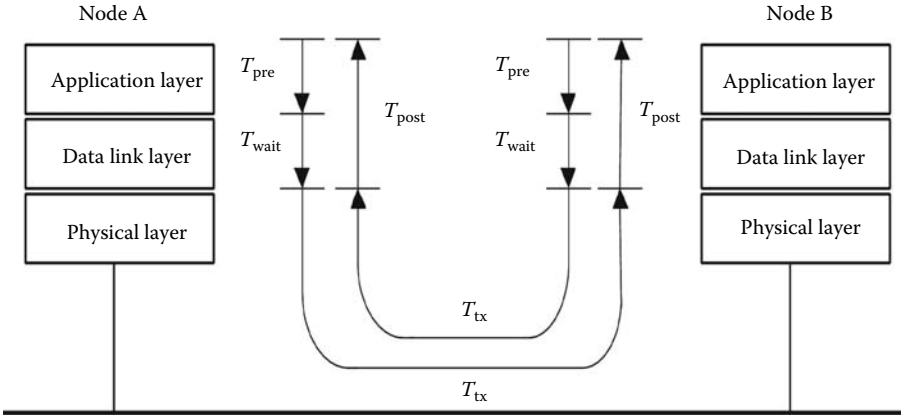


FIGURE 23.1 Timing diagram showing time spent sending a message from a source node to a destination node.

23.2.2.1 Pre- and Postprocessing Times

The preprocessing time at the source node depends on the device software and hardware characteristics. In many cases, it is assumed that the preprocessing time is constant or negligible. However, this assumption is not true in general; in fact, there may be noticeable differences in processing time characteristics between similar devices, and these delays may be significant. The postprocessing time at the destination node is the time taken to decode the network data into the physical data format and output it to the external environment.

In practical applications, it is very difficult to identify each individual timing component. However, a very straightforward experiment can be run with two nodes on the network. The source node A repeatedly requests data from a destination node B, and waits until it receives a response before sending another request. Because there are only two nodes on the network, there is never any contention, and thus the waiting time is zero. The request–response frequency is set low enough that no messages are queued up at the sender's buffer. The message traffic on the network is monitored, and each message is time-stamped. The processing time of each request–response pair, i.e., $T_{\text{post}} + T_{\text{pre}}$, can be computed by subtracting the transmission time from the time difference between the request and response messages. Because the time-stamps are recorded all at the same location, the problem of time synchronization across the network is avoided.

Figure 23.2 shows the experimentally determined device delays for DeviceNet devices utilizing the aforementioned setup [35,46]. Note that for all devices, the mean delay is significantly longer than the minimum frame time in DeviceNet (94 μs), and that the jitter is often significant. The uniform distribution of processing time at some of the devices is due to the fact that they have an internal sampling time, which is mismatched with the request frequency. Hence, the processing time recorded here is the sum of the actual processing time and the waiting time inside the device. The tested devices include photoeyes, input–output terminal blocks, mass flow controllers, and other commercially available DeviceNet devices.

A key point that can be taken from the data presented in Figure 23.2 is that the device processing time can be substantial in the overall calculation of T_{delay}. In fact, this delay often dominates over network delays. Thus, when designing industrial network systems to be used for control, device delay and delay variability should be considered as important factors when choosing the components. In the same manner, controller devices such as off-the-shelf PLCs typically specify scan times and interscan delays on the order of a few milliseconds, thus these controller delays can also dominate over network delays.

23.2.2.2 Waiting Time at Source Nodes

A message may spend time waiting in the queue at the sender's buffer and could be blocked from transmitting by other messages on the network. Depending on the amount of data, the source node must send and the traffic on the network, the waiting time may be significant. The main factors affecting waiting time are network protocol, message connection type, and network traffic load.

For control network operation, the message connection type must be specified. In a master–slave (MS) network,* there are three types of message connections: strobe, poll, and change of state (COS)/cyclic. In a "strobe" connection, the master device broadcasts a message to a group of devices and these devices respond with their current condition. In this case, all devices are considered to

* In this context, an MS network refers to operation from an end-to-end application layer perspective. Master node applications govern the method by which information is communicated to and from their slave node applications. Note that, as it will be described further in Section 23.3.1, application layer MS behavior does not necessarily require corresponding MS behavior at the MAC layer.

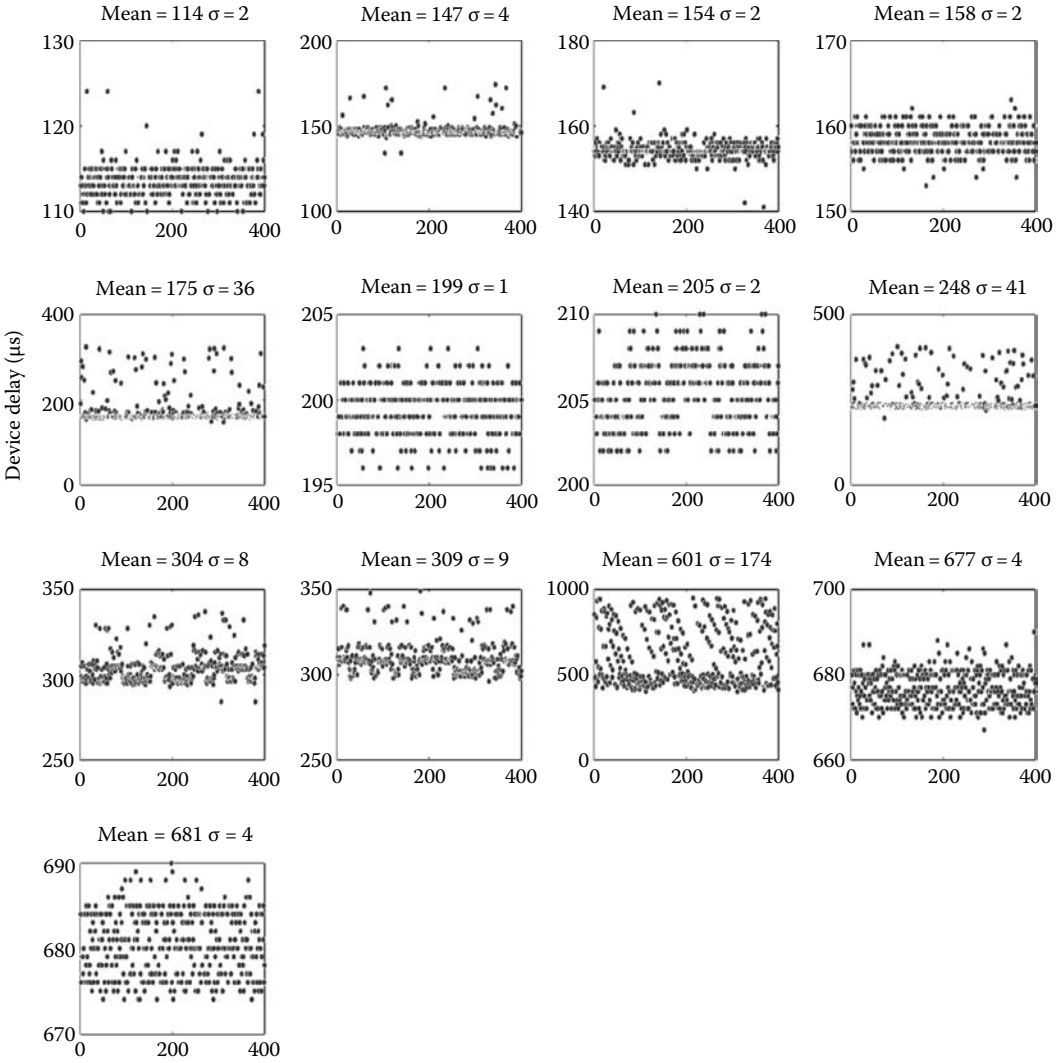

Mean = 114 σ = 2
Mean = 147 σ = 4
Mean = 154 σ = 2
Mean = 158 σ = 2
Mean = 175 σ = 36
Mean = 199 σ = 1
Mean = 205 σ = 2
Mean = 248 σ = 41
Mean = 304 σ = 8
Mean = 309 σ = 9
Mean = 601 σ = 174
Mean = 677 σ = 4
Mean = 681 σ = 4
Device delay (μs)

FIGURE 23.2 Device delays for DeviceNet devices in a request–response setup as described in Section 23.2.2.1. Delays are measured with only source and destination node communicating on the network and thus focus only on device delay jitter as described in Section 23.2.2. The stratification of delay times seen in some nodes is due to the fact that the smallest time that can be recorded is 1 μs.

sample new information at the same time. In a "poll" connection, the master sends individual messages to the polled devices and requests update information from them. Devices only respond with new signals after they have received a poll message. "COS/cyclic" devices send out messages either when their status is changed (COS) or periodically (cyclic). Although the COS/cyclic connection seems most appropriate from a traditional control systems point of view, strobe and poll are commonly used in industrial control networks [10].

For example, consider the strobe message connection in Figure 23.3. If Slave 1 is sending a message, the other eight devices must wait until the network medium is free. In a CAN-based DeviceNet network, it can be expected that Slave 9 will encounter the most waiting time because it has a lower priority on this priority-based network. However, in any network, there will be a nontrivial waiting time after a strobe, depending on the number of devices that will respond to the strobe.

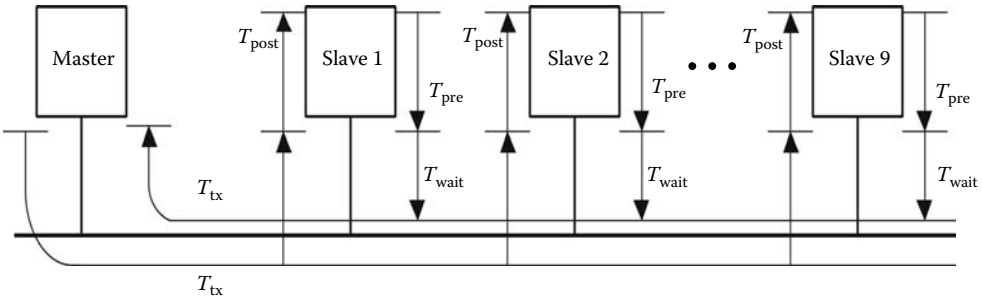


FIGURE 23.3 Waiting time diagram for a strobe message configuration.

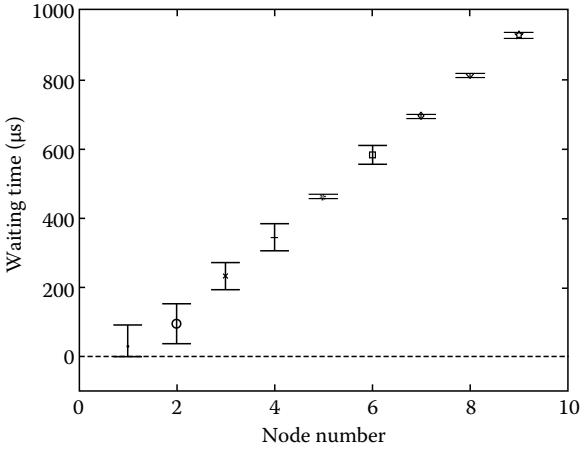


FIGURE 23.4 Nine identical devices with strobed message connection.

Figure 23.4 shows experimental data of the waiting time of nine identical devices with a strobed message connection on a DeviceNet network; 200 pairs of messages (request and response) were collected. Each symbol denotes the mean, and the distance between the upper and lower bars equals two standard deviations. If these bars are over the limit (maximum or minimum), then the value of the limit is used instead. It can be seen in Figure 23.4 that the average waiting time is proportional to the node number (i.e., priority). Although all these devices have a very low variance of processing time, the devices with the lowest node numbers have a larger variance of waiting time than the others, because the variance of processing time occasionally allows a lower priority device to access the idle network before a higher priority one.

23.2.3 Wired and Wireless QoS Metrics

The advent of wireless has added a new dimension to the QoS metric set that focuses heavily on the QoS of the communication medium with respect to external factors. With wired networks, reliability of data transmission is an important factor to consider. For example, some networks are physically more vulnerable than others to data corruption by electromagnetic interference. This issue is much more prevalent in wireless networks as there is a much higher potential exposure to external factors that can reduce the quality of the communication medium. Both wired and wireless networks use handshaking by sending of acknowledgment (ACK) messages to increase the reliability. If no ACK

message is received, the message is resent. These handshaking techniques increase the reliability of a network but also add to the required overhead and thus decrease the overall effective bandwidth.

Wireless networks also commonly utilize techniques such as frequency hopping and transmission on multiple frequency channels to make the transmission more robust to radio frequency (RF) interference sources such as other industrial wireless networks, cell phones, and manufacturing equipment such as spot welders which generate RF noise [7,72]. These frequency hopping protocols can be quite elaborate and often distinguish the robustness of one wireless technology over another.

Security is another factor that must be considered for both wired and wireless networks. For all networked systems, security is of special concern when networks and operating systems are used that can be vulnerable to Internet-based attacks and viruses [13]. Most industrial fieldbuses were not designed to be highly secure, relying mainly on physical isolation of the network instead of authentication or encryption techniques. When some type of security is provided, the intent is more commonly to prevent accidental misuse of process data than to thwart malicious network attacks [67]. For wireless, the security issue has another dimension in that the transmissions themselves can be intercepted rather easily by a listening device that understands the transmission protocol. The most common solution to this problem is encryption of the transmission through mechanisms such as virtual private network (VPN). While VPN provides for secure keyed "tunnels," mechanisms such as these also introduce transmission overhead and delay as shown in Figure 23.5. Thus the "cost of security" must be considered when evaluating the (especially wireless) networked solutions.

Other QoS and related metrics such as device power requirements, distances between nodes, and desire to "not" generate interference (i.e., coexist) with other networks can weigh heavily in the choice of a wireless protocol and solution [7]. The key to evaluating solutions against metrics is to be able to quantify all of the important metrics and evaluate them collectively in the application environment. This process is discussed in Section 23.4.

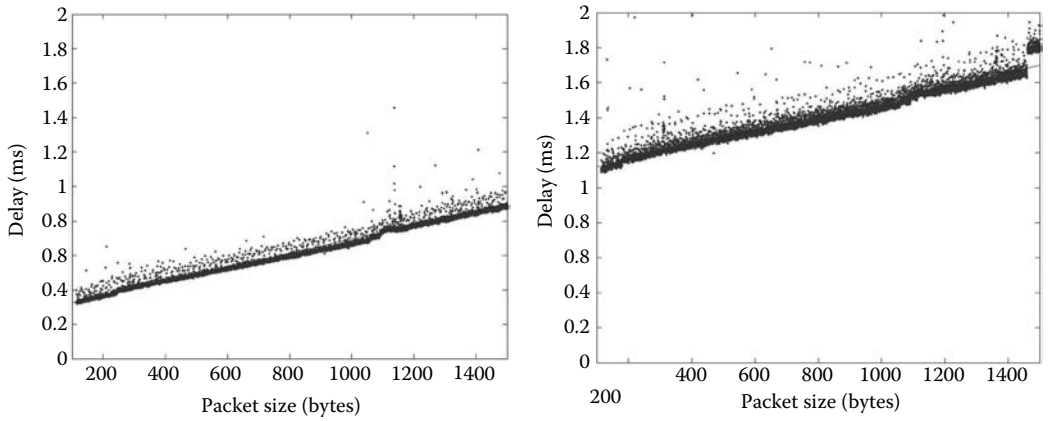


FIGURE 23.5 Example of the impact of VPN security encryption on Ethernet network performance (the figure on the left is the round-trip delay measured for user datagram protocol (UDP)* packet transmission on a 100 Mbps switched Ethernet network; the figure on the right is the delay measured for the same system, but with VPN encryption added).

* UDP is a relatively fast unacknowledged communication service utilized in Ethernet, often utilized as a baseline in defining Ethernet system performance.

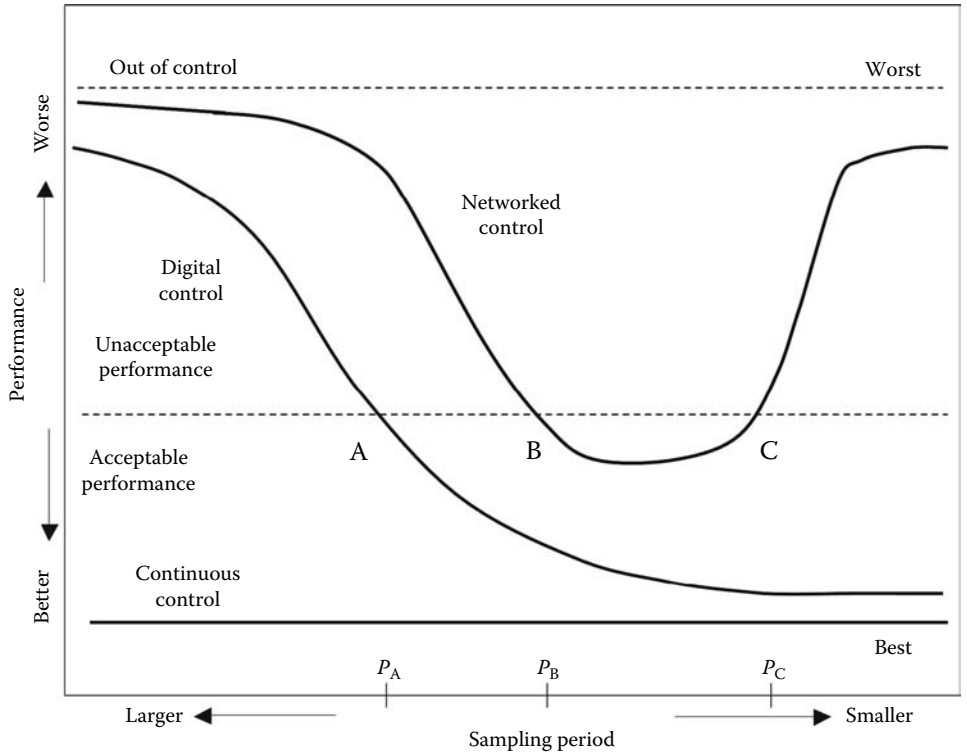


FIGURE 23.6 Performance comparison of continuous control, digital control, and networked control, as a function of sampling frequency.

23.2.4 Network QoS vs. System Performance

When a network is used in the feedback loop of a control system, the performance of the system depends not only on the QoS of the network, but also on how the network is used (e.g., sample time, message scheduling, node prioritization, etc.) [35,37]. For example, consider a continuous-time control system that will be implemented with networked communication. Figure 23.6 shows how the control performance varies vs. sampling period in the cases of continuous control, digital control, and networked control. The performance of the continuous control system is independent of the sample time (for a fixed control law). The performance of the digital control system approaches the performance of the continuous-time system as the sampling frequency increases [22]. In an NCS, the performance is worse than the digital control system at low frequencies, due to the extra delay associated with the network (as described in Section 23.2.2). Also, as the sampling frequency increases, the network starts to become saturated, data packets are lost or delayed, and the control performance rapidly degrades. Between these two extremes lies a "sweet spot" where the sample period is optimized to the control and networking environment. Note that, as discussed in Section 23.2.2, the device delay can comprise a significant portion of the end-to-end delay; thus, the optimal sampling period can often be more a function of the device speed than the network speed, especially for faster networks with minimal collisions, such as switched Ethernet.

Typical performance criteria for feedback control systems include overshoot to a step reference, steady-state tracking error, phase margin, or time-averaged tracking error [21]. The performance criteria in Figure 23.6 can be one of these or a combination of them. Due to the interaction of the

network and control requirements, the selection of the best sampling period is a compromise. More details on the performance computation and analysis of points A, B, and C in Figure 23.6 can be found in Ref. [35], including simulation and experimental results that validate the overall shape of the chart.

23.3 Differentiation of Industrial Networks

Networks can be differentiated either by their protocol (at any or all levels of the ISO–OSI seven-layer reference model [27]) or by their primary function (control, diagnostics, and safety). These dimensions of differentiation are somewhat related. In this section, we first define how network protocols are categorized technically with respect to timing, and then discuss the different types of protocols that are commonly used in industrial networks. In Section 23.5, we describe how these different types of networks are used for different functions.

23.3.1 Categorization of Networks

When evaluating the network QoS parameters associated with timeliness, determinism, etc., the protocol functionality at the data link layer is the primary differentiator among network protocol types. Specifically, the MAC sublayer protocol within the data link layer describes the protocol for obtaining access to the network. The MAC sublayer thus is responsible for satisfying the time-critical/real-time response requirement over the network and for the quality and reliability of the communication between network nodes [30]. The discussion, categorization, and comparison in this section thus focus on the MAC sublayer protocols.

There are three main types of medium access control used in control networks: time-division multiplexing (such as MS or token-passing [TP]), RA with retransmission when collisions occur (e.g., Ethernet and most wireless mechanisms), and RA with prioritization for collision arbitration (e.g., CAN). Implementations can be hybrids of these types; for example, switched Ethernet combines time-division multiplexed (TDM) and RA. Note that, regardless of the MAC mechanism, most network protocols support some form of MS communication at the application level; however, this appearance of TDM at the application level does not necessarily imply the same type of parallel operation at the MAC level. Within each of these three MAC categories, there are numerous network protocols that have been defined and used.

A recent survey of the types of control networks used in industry shows a wide variety of networks in use; see Table 23.1 and also Refs. [23,36,66]. The networks are classified according to type: RA with CD, CA, or arbitration on message priority (AMP); or TDM using TP or MS.

TABLE 23.1 Most Popular Fieldbuses

Network	Type	Users (%)	Max. Speed	Max. Devices
Ethernet TCP/IP	RA/CD	78	1 Gb/s	1024
Modbus	TDM/MS	48	35 Mb/s	32
DeviceNet	RA/AMP	47	500 kb/s	64
ControlNet	TDM/TP	39	5 Mb/s	99
WiFi (IEEE 802.11b)	RA/CA	35	11 Mb/s	not specified
Modbus TCP	TDM/MS	34	1 Gb/s	256
PROFIBUS-DP	TDM/MS and TP	27	12 Mb/s	127
AS-I	TDM/MS	17	167 kb/s	31

Source: Grid Connect. The Grid Connect Fieldbus comparison chart. http://www.synergetic.com/compare.htm; Montague, J., *Control Eng.*, 52(3), 2005.

Note: The maximum speed depends on the physical layer, not the application-level protocol. Note that the totals are more than 100% because most companies use more than one type of bus.

23.3.2 Time-Division Multiplexing

Time-division multiplexing can be accomplished in one of two ways: MS or TP. In an MS network, a single master polls multiple slaves. Slaves can only send data over the network when requested by the master; there are no collisions, as the data transmissions are carefully scheduled by the master. A TP network has multiple masters, or peers. The token bus protocol (e.g., IEEE 802.4) allows a linear, multidrop, tree-shaped, or segmented topology [71]. The node that currently has the token is allowed to send data. When it is finished sending data, or the maximum token holding time has expired, it "passes" the token to the next logical node on the network. If a node has no message to send, it just passes the token to the successor node. The physical location of the successor is not important because the token is sent to the logical neighbor. Collision of data frames does not occur, as only one node can transmit at a time. Most TP protocols guarantee a maximum time between network accesses for each node, and most also have provisions to regenerate the token if the token holder stops transmitting and does not pass the token to its successor. AS-I, Modbus, and Interbus-S are typical examples of MS networks, while PROFIBUS and ControlNet are typical examples of TP networks. Each peer node in a PROFIBUS network can also behave like a master and communicate with a set of slave nodes during the time it holds the token [56].

TP networks are deterministic because the maximum waiting time before sending a message frame can be characterized by the token rotation time. At high utilizations, TP networks are very efficient and fair. There is no time wasted on collisions, and no single node can monopolize the network. At low utilizations, they are inefficient due to the overhead associated with the TP protocol. Nodes without any data to transmit must still receive and pass the token.

Waiting time in a TDM network can be determined explicitly once the protocol and the traffic to be sent on the network are known. For TP networks, the node with data to send must first wait to receive the token. The time it needs to wait can be computed by adding up the transmission times for all of the messages on nodes ahead of it in the logical ring. For example, in ControlNet, each node holds the token for a minimum of 22.4 μs and a maximum of 827.2 μs.

In MS networks, the master typically polls all slaves every cycle time. Slaves cannot transmit data until they are polled. After they are polled, there is no contention for the network so the waiting time is zero. If new data are available at a slave (e.g., a limit switch trips), the slave must wait until it is polled before it can transmit its information. In many MS networks (such as AS-Interface), the master will only wait for a response from a slave until a timer has expired. If the slave does not respond within the timeout value for several consecutive polls, it is assumed to have dropped off the network. Also, every cycle time, the master attempts to poll an inactive slave node (in a round-robin fashion) [4]. In this way, new slaves can be added to the network and will be eventually noticed by the master.

23.3.3 Random Access with Collision Arbitration: CAN (CSMA/AMP)

CAN is not only a serial communication protocol developed mainly for applications in the automotive industry, but also capable of offering good performance in other time-critical industrial applications. The CAN protocol is optimized for short messages and uses a carrier sense multiple access (CSMA)/AMP medium access method. Thus, the protocol is message-oriented, and each message has a specific priority that is used to arbitrate access to the bus in case of simultaneous transmission. The bit stream of a transmission is synchronized on the start bit, and the arbitration is performed on the following message identifier, in which a logic zero is dominant over a logic one. A node that wants to transmit a message waits until the bus is free and then starts to send the identifier of its message bit by bit. Conflicts for access to the bus are solved during transmission by an arbitration process at the bit level of the arbitration field, which is the initial part of each frame. Hence, if two devices want to send messages at the same time, they first continue to send the message frames and then listen to the network. If one of them receives a bit different from the one it sends out, it loses

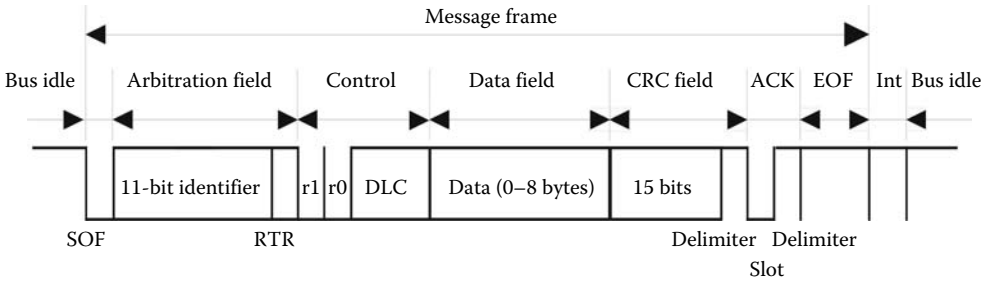


FIGURE 23.7 Message frame format of DeviceNet (standard CAN format).

the right to continue to send its message, and the other wins the arbitration. With this method, an ongoing transmission is never corrupted, and collisions are nondestructive [33].

DeviceNet is an example of a technology based on the CAN specification that has received considerable acceptance in device-level manufacturing applications. The DeviceNet specification is based on the standard CAN with an additional application and physical layer specification [10,33].

The frame format of DeviceNet is shown in Figure 23.7 [10]. The total overhead is 47 bits, which includes start of frame (SOF), arbitration (11-bit identifier), control, cyclic redundancy check (CRC), ACK, end of frame (EOF), and intermission (INT) fields. The size of a data field is between 0 and 8 bytes. The DeviceNet protocol uses the arbitration field to provide source and destination addressing as well as message prioritization.

The major disadvantage of CAN compared with the other networks is the slow data rate, limited by the network length. Because of the bit-synchronization, the same data must appear at both ends of the network simultaneously. DeviceNet has a maximum data rate of 500 kb/s for a network of 100 m. Thus, the throughput is limited compared with other control networks. CAN is also not suitable for transmission of messages of large data sizes, although it does support fragmentation of data that is more than 8 bytes into multiple messages.

23.3.4 Ethernet-Based Networks

The proliferation of the Internet has led to the pervasiveness of Ethernet in both homes and businesses. Because of its low cost, widespread availability, and high communication rate, Ethernet has been proposed as the ideal network for industrial automation [9,53]. Some question whether Ethernet will become the de facto standard for automation networks, making all other solutions obsolete [19,63]. However, standard Ethernet (IEEE 802.3) is not a deterministic protocol, and network QoS cannot be guaranteed [9,33]. Collisions can occur on the network, and messages must be retransmitted after random amounts of time. To address this inherent nondeterminism, many different "flavors" of Ethernet have been proposed for use in industrial automation. Several of these add layers on top of standard Ethernet or on top of the TCP/IP protocol suite to enable the behavior of Ethernet to be more deterministic [17,19,31]. In this way, the network solutions may no longer be "Ethernet" other than at the physical layer; they may use the same hardware but are not interoperable. As noted in Ref. [36], message transmission does not always lead to successful communication: "just because you can make a telephone ring in Shanghai doesn't mean you can speak Mandarin." A more effective and accepted solution in recent years has been the utilization of switches to manage the Ethernet bandwidth utilizing a TDM approach among time-critical nodes. Rather than repeat the survey of current approaches to industrial Ethernet in Ref. [17], in this section, the general MAC protocol of Ethernet is outlined, and the general approaches that are used with Ethernet for industrial purposes are discussed.

Ethernet is an RA network, also often referred to as CSMA. Each node listens to the network and can start transmitting at any time that the network is free. Typically, once the network is clear, a node must wait for a specified amount of time (the interframe time) before sending a message. To reduce collisions on the network, nodes wait an additional random amount of time called the backoff time before they start transmitting. Some types of messages (e.g., MAC layer ACKs) may be sent after a shorter interframe time. Priorities can be implemented by allowing for shorter interframe times for higher priority traffic. However, if two nodes start sending messages at the exact same time (or if the second node starts transmitting before the first message arrives at the second node), there will be a collision on the network. Collisions in Ethernet are destructive; the data are corrupted and the messages must be resent.

There are three common flavors of Ethernet: (1) hub-based Ethernet, which is common in office environments and is the most widely implemented form of Ethernet, (2) switched Ethernet, which is more common in manufacturing and control environments, and (3) wireless Ethernet. Each of these is discussed in detail below.

23.3.4.1 Hub-Based Ethernet (CSMA/CD)

Hub-based Ethernet uses hub(s) to interconnect the devices on a network; this type of Ethernet is common in the office environment. When a packet comes into one hub interface, the hub simply broadcasts the packet to all other hub interfaces. Hence, all of the devices on the same network receive the same packet simultaneously, and message collisions are possible. Collisions are dealt with utilizing the CSMA/CD protocol as specified in the IEEE 802.3 network standard [5,8,65].

This protocol operates as follows: when a node wants to transmit, it listens to the network. If the network is busy, the node waits until the network is idle; otherwise, it can transmit immediately (assuming an interframe delay has elapsed since the last message on the network). If two or more nodes listen to the idle network and decide to transmit simultaneously, the messages of these transmitting nodes collide and the messages are corrupted. While transmitting, a node must also listen to detect a message collision. On detecting a collision between two or more messages, a transmitting node transmits 32 jam bits and waits a random length of time to retry its transmission. This random time is determined by the standard binary exponential backoff (BEB) algorithm: the retransmission time is randomly chosen between 0 and (2^i) slot times, where i denotes the ith collision event detected by the node and one slot time is the minimum time needed for a round-trip transmission. However, after 10 collisions have been reached, the interval is fixed at a maximum of 1023 slots. After 16 collisions, the node stops attempting to transmit and reports failure back to the node microprocessor. Further recovery may be attempted in higher layers [65].

The Ethernet frame format is shown in Figure 23.8 [65]. The total overhead is 26 $(= 22 + 4)$ bytes. The data packet frame size is between 46 and 1500 bytes. There is a nonzero minimum data size requirement because the standard states that valid frames must be at least 64 bytes long, from

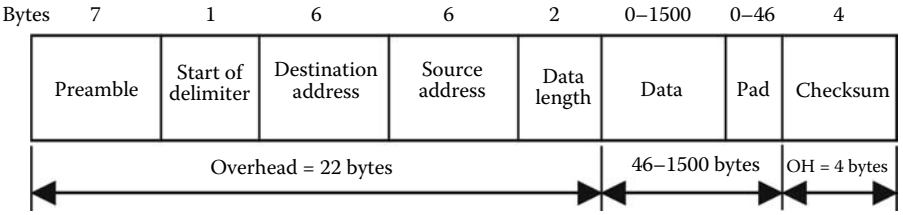


FIGURE 23.8 Ethernet (CSMA/CD) frame format; 20 Byte interframe space not shown.

destination address to checksum (72 bytes including preamble and start of delimiter). If the data portion of a frame is less than 46 bytes, the pad field is used to fill out the frame to the minimum size. There are two reasons for this minimum size limitation. First, it makes it easier to distinguish valid frames from "garbage." Because of frame truncation, stray bits and pieces of frames frequently appear on the cable. Second, it prevents a node from completing the transmission of a short frame before the first bit has reached the far end of the cable, where it may collide with another frame. For a 10-Mbps Ethernet with a maximum length of 2500 m and four repeaters, the minimum allowed frame time or slot time is 51.2 μs, which is the time required to transmit 64 bytes at 10 Mbps [65].

Because of low medium access overhead, Ethernet uses a simple algorithm for operation of the network and has almost no delay at low network loads [71]. No communication bandwidth is used to gain access to the network compared with TP protocols. However, Ethernet is a nondeterministic protocol and does not support any message prioritization. At high network loads, message collisions are a major problem because they greatly affect data throughput and time delays may become unbounded [71]. The Ethernet "capture" effect existing in the standard BEB algorithm, in which a node transmits packets exclusively for a prolonged time despite other nodes waiting for medium access, causes unfairness and substantial performance degradation [57]. Based on the BEB algorithm, a message may be discarded after a series of collisions; therefore, end-to-end communication is not guaranteed. Because of the required minimum valid frame size, Ethernet uses a large message size to transmit a small amount of data.

Several solutions have been proposed for using this form of Ethernet in control applications [9]. For example, every message could be time-stamped before it is sent. This requires clock synchronization, however, which has not traditionally been easy to accomplish [15], although the IEEE 1588 standard has recently emerged to enable clock synchronization over networks [26]. Various schemes based on deterministic retransmission delays for the collided packets of a CSMA/CD protocol result in an upper-bounded delay for all the transmitted packets. However, this is achieved at the expense of inferior performance to CSMA/CD at low-to-moderate channel utilization in terms of delay throughput [30]. Other solutions also try to prioritize CSMA/CD (e.g., LonWorks) to improve the response time of critical packets [47]. To a large extent these solutions have been rendered moot with the proliferation of switched Ethernet as described below. On the other hand, many of the same issues reappear with the migration to wireless Ethernet for control.

23.3.4.2 Switched Ethernet (CSMA/CA)

Switched Ethernet utilizes switches to subdivide the network architecture, thereby avoiding collisions, increasing network efficiency, and improving determinism. It is widely used in manufacturing applications. The main difference between switch- and hub-based Ethernet networks is the intelligence of forwarding packets. Hubs simply pass on incoming traffic from any port to all other ports, whereas switches learn the topology of the network and forward packets to the destination port only. In a starlike network layout, every node is connected with a single cable to the switch as a full-duplex point-to-point link. Thus, collisions can no longer occur on any network cable. Switched Ethernet relies on this star cluster layout to achieve this collision-free property.

Switches employ the cut-through or store-and-forward technique to forward packets from one port to another, using per-port buffers for packets waiting to be sent on that port. Switches with cut-through first read the MAC address and then forward the packet to the destination port according to the MAC address of the destination and the forwarding table on the switch. On the other hand, switches with store-and-forward examine the complete packet first. Using the CRC code, the switch will first verify that the frame has been correctly transmitted before forwarding the packet to the destination port. If there is an error, the frame will be discarded. Store-and-forward switches are slower, but will not forward any corrupted packets.

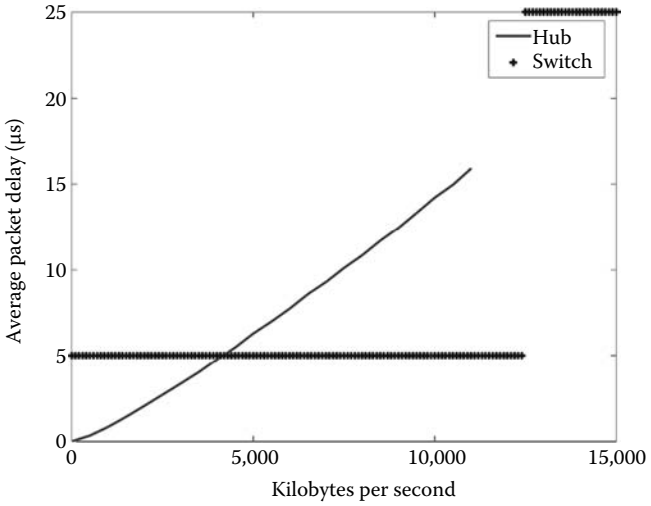


FIGURE 23.9 Packet delay as a function of node traffic for a hub and a switch [48]. Simulation results with baselines (delay magnitudes) computed from experiments.

Although there are no message collisions on the networks, congestion may occur inside the switch when one port suddenly receives a large number of packets from the other ports. If the buffers inside the switch overflow, messages will be lost [17]. Three main queuing principles are implemented inside the switch in this case. They are first-in-first-out (FIFO) queue, priority queue, and per-flow queue. The FIFO queue is a traditional method that is fair and simple. However, if the network traffic is heavy, the network QoS for timely and fair delivery cannot be guaranteed. In the priority queuing scheme, the network manager reads some of the data frames to distinguish which queues will be more important. Hence, the packets can be classified into different levels of queues. Queues with high priority will be processed first followed by queues with low priority until the buffer is empty. With the per-flow queuing operation, queues are assigned different levels of priority (or weights). All queues are then processed one by one according to priority; thus, the queues with higher priority will generally have higher performance and could potentially block queues with lower priority [9].

Thus, although switched Ethernet can avoid the extra delays due to collisions and retransmissions, it can introduce delays associated with buffering and forwarding. This trade-off can be seen in Figure 23.9, which shows the average packet delay as a function of node traffic. The switch delay is small but constant until the buffer saturates and packets must be resent; the hub delay increases more gradually. Examples of timing analysis and performance evaluation of switched Ethernet can be found in Refs. [32,48,54,69].

23.3.4.3 Industrial Ethernet

In an effort to package Ethernet to be more suitable for industrial applications, a number of "industrial Ethernet" protocols have emerged. These include EtherNet/IP, Modbus/TCP, and PROFINET. While these protocol specifications vary to some extent at all levels of the OSI model, they all fundamentally utilize or recommend switched Ethernet technology as defined in the previous subsection. Thus, the differences in performance between Industrial Ethernet technologies lie more with the devices than the protocols. For example in an effort to understand the trade-offs between Industrial Ethernet technologies, two common Industrial Ethernet protocols, EtherNet/IP and PROFINET, were compared in the areas of architecture principles, technologies incorporated, performance, ease of use, diagnostics

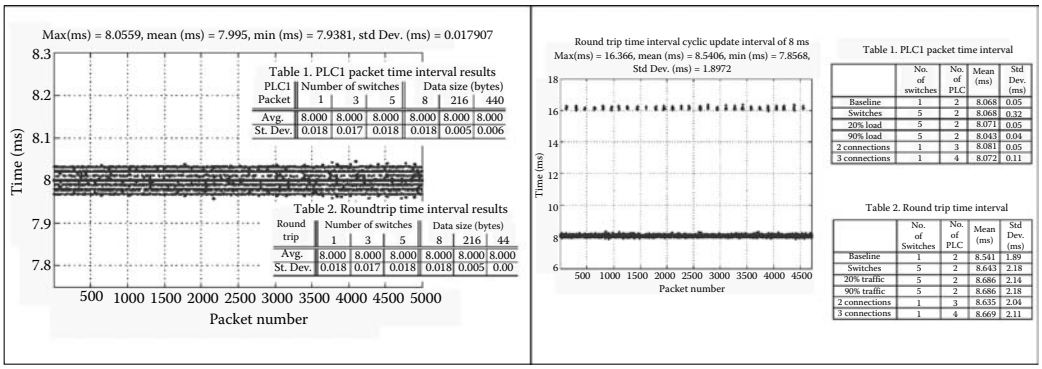

Max(ms) = 8.0559, mean (ms) = 7.995, min (ms) = 7.9381, std Dev. (ms) = 0.017907
Time (ms)
Packet number
Table 1. PLC1 packet time interval results
PLC1 Packet | Number of switches 1 3 5 | Data size (bytes) 8 216 440
Avg. 8.000 8.000 8.000 8.000 8.000 8.000
St. Dev. 0.018 0.017 0.018 0.018 0.005 0.006
Table 2. Roundtrip time interval results
Round trip | Number of switches 1 3 5 | Data size (bytes) 8 216 44
Avg. 8.000 8.000 8.000 8.000 8.000 8.000
St. Dev. 0.018 0.017 0.018 0.018 0.005 0.00
Round trip time interval cyclic update interval of 8 ms
Max(ms) = 16.366, mean (ms) = 8.5406, min (ms) = 7.8568,
Std Dev. (ms) = 1.8972
Time (ms)
Packet number
Table 1. PLC1 packet time interval
No. of switches | No. of PLC | Mean (ms) | Std Dev. (ms)
Baseline 1 2 8.068 0.05
Switches 5 2 8.068 0.32
20% load 5 2 8.071 0.05
90% load 5 2 8.043 0.04
2 connections 1 3 8.081 0.05
3 connections 1 4 8.072 0.11
Table 2. Round trip time interval
No. of Switches | No. of PLC | Mean (ms) | Std Dev. (ms)
Baseline 1 2 8.541 1.89
Switches 5 2 8.643 2.18
20% traffic 5 2 8.686 2.14
90% traffic 5 2 8.686 2.18
2 connections 1 3 8.635 2.04
3 connections 1 4 8.669 2.11

FIGURE 23.10 Sample plots of round-trip timing measurements for PROFINET (5a) and EtherNet/IP (5b). Here, the round-trip times for packets between two PLCs are plotted for a large number of packets to obtain a time distribution. The embedded tables represent the consolidation of a number of these graphs, where the testing environment is modified in terms of switches between sender and receiver, data size transmitted, and network loading (the plots shown represent the "Baseline" case).

capabilities, and network management capabilities* [43]. As part of this effort, parallel multilayer switched Ethernet testbeds were developed utilizing each of these technologies, where the network layout was representative of the structure being utilized at a leading automotive manufacturer.

The results indicate that both protocols and protocol devices are fairly similar and are adequate to the task of providing industrial networking capabilities at the PLC level and higher [43]. However, distinct differences were observed, such as those illustrated in Figure 23.10, that indicate additional improvements in device performance may be needed if the solution is to be deployed down to the I/O level.

23.3.5 Impact of Ethernet Application Layer Protocols: OPC

OPC is an open communication standard that is often used in industry to connect SCADA systems and human–machine interfaces (HMIs) to control systems and fieldbus networks [24,38,62]†. It is based on the Microsoft DCOM standard [51] and is the dominant factory-floor application layer protocol utilized for diagnostics and is beginning to be used for sequential control [50]. The main benefit of OPC is that it allows any products that support the standard to share data. Although OPC actually consists of many different communication specifications, its most commonly used form is called Data Access, which supports both client–server and publisher–subscriber communication models. The server maintains a table of data values, and the client can read or write updates. The overhead associated with OPC (and DCOM in general) is significant, as shown in Figure 23.11. Most of this delay is due to the software implementation of the OPC protocol; OPC was never intended for a real-time environment. However, it is very useful to push data up from the low-level controls to the higher-level supervisors or diagnostic systems. It can also be used to send commands down from the HMIs to the control systems. Its high level of interoperability enables the connection of multiple control systems from different vendors in a unified manner. However, when OPC is used to send control data, the additional delay caused by the higher-level application layer protocol must be considered.

* Many industrial Ethernet protocols are available, with Modbus/TCP the most widely utilized [23,40], however in this instance the manufacturer was interested in comparing these two industrial Ethernet protocols.

† While OPC originally stood for "OLE for Process Control," the official stance of the OPC Foundation is that OPC is no longer an acronym and the technology is simply known as OPC.

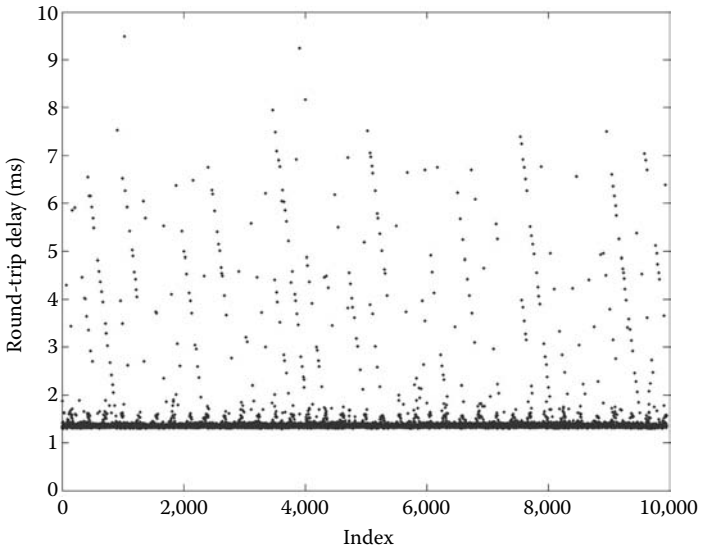


FIGURE 23.11 Illustration of overhead delay associated with OPC over a 100 Mb/s switched Ethernet network. Mean = 1.5 ms with standard deviation = 0.03 ms; for comparison purposes the mean UDP round-trip time over the same network is 0.33 ms.

TABLE 23.2 High-Level Description of Common Wireless Technologies Illustrating Trade-Offs

Characteristic	802.11	Bluetooth	ZigBee
Range	High	Medium, depends on class	Low
Power consumption	High	Medium	Low
Data rate	Very high	High	Low
Optimal data size	High	Medium	Low
Redundancy	Low	Low	High

23.3.6 Wireless Networks

The move to wireless has resulted in yet another explosion of protocol technologies due to the fact that the change in the media from a wire to air has resulted in the need for higher level of consideration of many external issues (e.g., noise, coexistence, and security) as well as the opportunity for development of more elaborate protocols to address these issues (e.g., frequency hopping, dynamic transmission power algorithms, and new CA and backoff-style algorithms) [7]. Table 23.2 provides a high-level comparison of three technologies that are utilized in manufacturing that offer somewhat complementary capabilities. From the table it is clear that, at least at this time, one size does not fit all when it comes to matching a wireless protocol to application requirements. In the remainder of this section, 802.11 and Bluetooth are explored in detail. For additional discussion of ZigBee see Refs. [7,73].

23.3.6.1 Wireless Ethernet (CSMA/CA)

Wireless Ethernet, based on the IEEE 802.11 standard, can replace wired Ethernet in a transparent way as it implements the two lowest layers of the ISO–OSI model [27,72]. Besides the physical layer, the biggest difference between 802.11 and 802.3 is in the medium access control. Unlike wired Ethernet nodes, wireless stations cannot "hear" a collision. A CA mechanism is used but cannot entirely

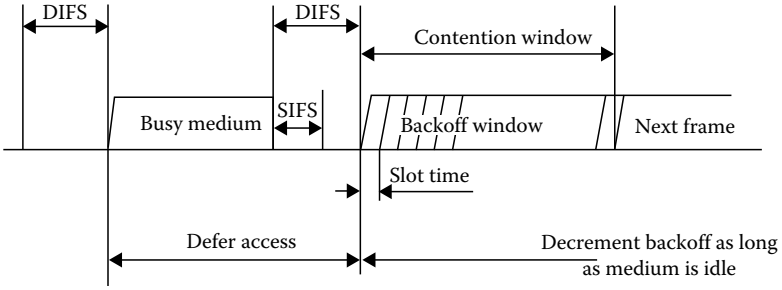


FIGURE 23.12 Timing diagram for wireless Ethernet (IEEE 802.11).

prevent collisions. Thus, after a packet has been successfully received by its destination node, the receiver sends a short ACK packet back to the original sender. If the sender does not receive an ACK packet, it assumes that the transmission was unsuccessful and retransmits.

The CA mechanism in 802.11 works as follows. If a network node wants to send while the network is busy, it sets its backoff counter to a randomly chosen value. Once the network is idle, the node waits first for an interframe space (DIFS) and then for the backoff time before attempting to send (see Figure 23.12). If another node accesses the network during that time, it must wait again for another idle interval. In this way, the node with the lowest backoff time sends first. Certain messages (e.g., ACK) may start transmitting after a shorter interframe space (SIFS), thus they have a higher priority. Collisions may still occur because of the random nature of the backoff time; it is possible for two nodes to have the same backoff time.

Several refinements to the protocol also exist. Nodes may reserve the network either by sending a request to send message or by breaking a large message into many smaller messages (fragmentation); each successive message can be sent after the smallest interframe time. If there is a single master node on the network, the master can poll all the nodes and effectively create a TDM contention-free network.

In addition to time delays, the difference between the theoretical data rate and the practical throughput of a control network should be considered. For example, raw data rates for 802.11 wireless networks range from 11 to 54 Mbits/s. The actual throughput of the network, however, is lower due to both the overhead associated with the interframe spaces, ACK, and other protocol support transmissions, and to the actual implementation of the network adapter. Although 802.11a and 802.11g have the same raw data rate, the throughput is lower for 802.11g because its backward compatibility with 802.11b requires that the interframe spaces be as long as they would be on the 802.11b network. Computed and measured throughputs are shown in Table 23.3 [12]. The experiments were conducted by continually sending more traffic on the network until a further setpoint increase in traffic resulted in no additional throughput.

Experiments conducted to measure the time delays on wireless networks are summarized in Table 23.4 and Figure 23.7 [12]. Data packets were sent from the client to the server and back again, with varying amounts of cross-traffic on the network. The send and receive times on both machines were time-stamped. The packet left the client at time t_a and arrived at the server at time t_b; then left

TABLE 23.3 Maximum Throughputs for Different 802.11 Wireless Ethernet Networks

Network Type	802.11a	802.11g	802.11b
Nominal data rate	54	54	11
Theoretical throughput	26.46	17.28	6.49
Measured throughput	23.2	13.6	3.6

Note: All data rates and throughputs are in Mb/s.

TABLE 23.4 Computed Frame Times and Experimentally Measured Delays on Wireless Networks

Network Type	802.11a	802.11g	802.11b
Frame time (UDP), computed	0.011	0.011	0.055
Median delay (UDP), measured	0.346	0.452	1.733
Frame time (OPC), computed	0.080	0.080	0.391
Median delay (OPC), measured	2.335	2.425	3.692

Note: All times in ms.

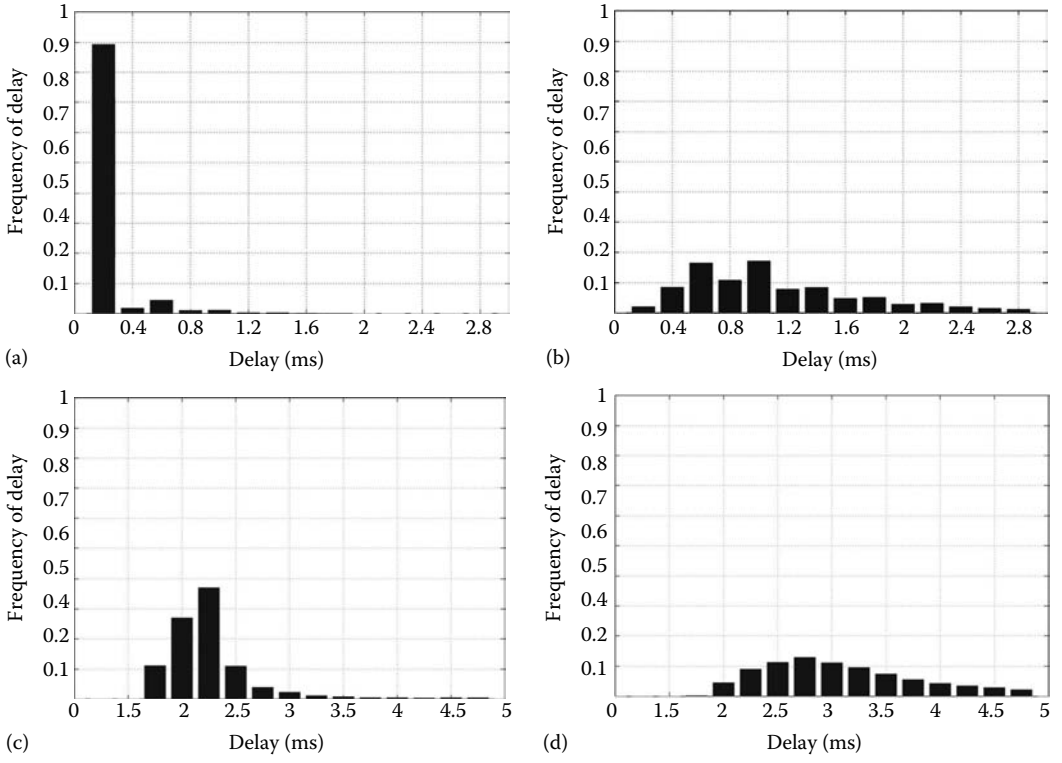


FIGURE 23.13 Distributions of packet delays for different values of cross-traffic throughput on an 802.11a network. (a) UDP delays, 3Mb/s cross-traffic, (b) UDP delays, 22Mb/s cross-traffic, (c) OPC delays, 3Mb/s cross-traffic, and (d) OPC delays, 22Mb/s cross-traffic.

the server at time t_c and arrived at the client at time t_d. The sum of the pre- and postprocessing times and the transmission time on the network for both messages can be computed as (assuming that the two nodes are identical)

$$2 * T_{\text{delay}} = 2 * \left(T_{\text{pre}} + T_{\text{wait}} + T_{\text{tx}} + T_{\text{post}}\right) = t_d - t_a - \left(t_c - t_b\right)$$

Note that this measurement does not require that the clocks on the client and server be synchronized. As the delays at the two nodes can be different, it is this sum of the two delays that is plotted in Figure 23.13 and tabulated in Table 23.4.

Two different types of data packets were considered: user datagram protocol (UDP) and OPC. UDP packets carry only a data load of 50 bytes. OPC requires extra overhead to support an application layer; consequently, the OPC packets contain 512 data bytes (in addition to the overhead). For comparison purposes, the frame times (including the overheads) are computed for the different packets.

23.3.6.2 Bluetooth

Bluetooth was originally designed as a wireless replacement for cellular phone headsets; however, its mid-range wireless capabilities in terms of distance and power consumption, combined with its frequency hopping capability for improved noise immunity, have led to its consideration as a wireless solution for lower-level connectivity on the manufacturing floor. Bluetooth operates in the same 2.4 GHz frequency band as 802.11b and g, and defines a full OSI communications stack, with the lower levels published as IEEE standard 802.15.1 [6]. Advantages of Bluetooth which make it ideal for manufacturing include dynamic connection establishment without need for human interaction, utilization of forward error correction (FEC) for delivering the messages without error and without requiring retransmission, and a frequency hopping capability for improved noise immunity and coexistence that includes a capability to "learn" of and avoid areas in the spectrum that are crowded with other transmissions. Drawbacks include loss of efficiency from the FEC capability (a 1 Mbps communications channel can deliver only 721 Kbps), and disruption of 802.11 transmissions (coexistence issue) due to the fact that the technology frequency hops at a faster rate than specified for 802.11b/g [6,7].

Bluetooth applications have also been shown to exhibit nondeterministic response time behavior in the face of ambient interference. While this behavior is not limited to Bluetooth wireless systems, it provides a good example of an issue that must be addressed before common wireless technologies can be utilized effectively in NCSs. As an example, Figure 23.13 shows the results of an experiment to better understand the determinism of a Bluetooth NCS. As shown in Figure 23.14a, the test setup includes two Bluetooth nodes, namely, a controller and an I/O block, and a DeviceNet network

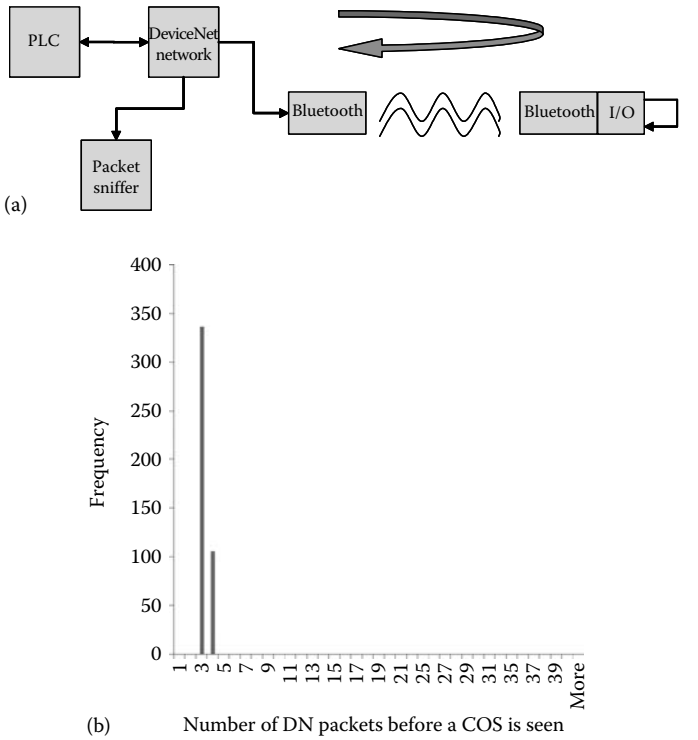
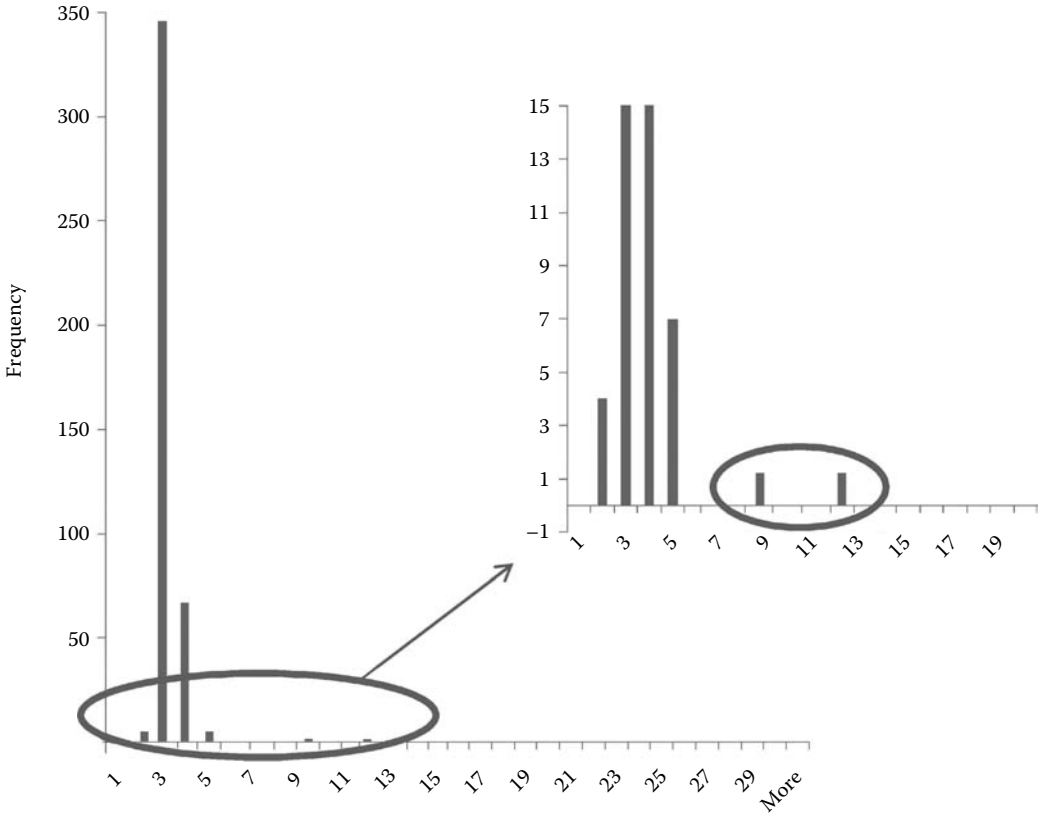
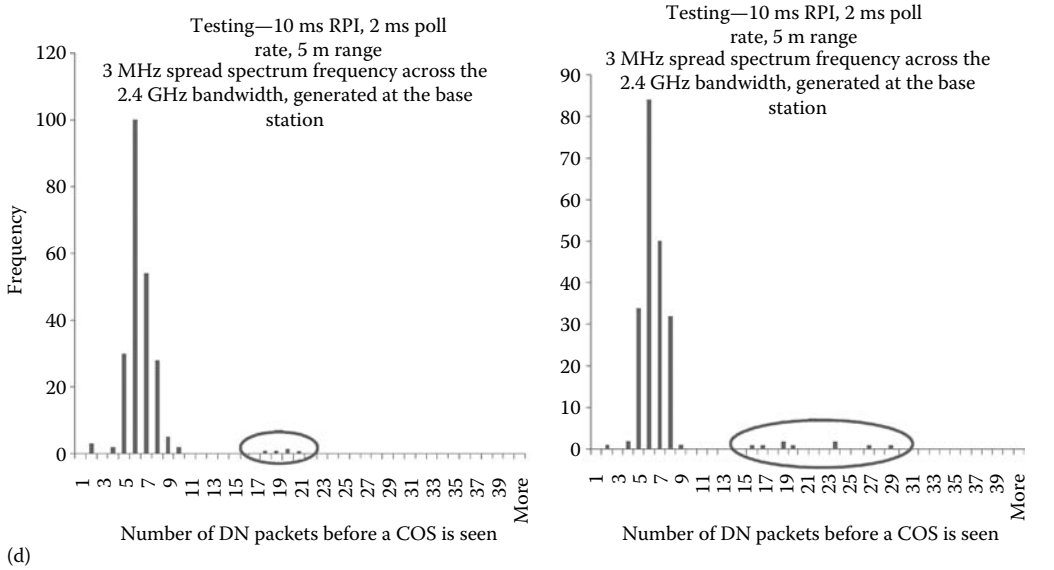


FIGURE 23.14 Experimental characterization of determinism in a Bluetooth system. (a) Bluetooth determinism performance experimental setup, (b) baseline result with minimal channel noise and near ideal distance between nodes, (c) distance test (10 m) is inset expanded view of nondeterminism, and (d) interference test, with nondeterminism highlighted.

(*continued*)



(c)


Testing—10 ms RPI, 2 ms poll rate, 5 m range
3 MHz spread spectrum frequency across the 2.4 GHz bandwidth, generated at the base station
Frequency
120
100
80
60
40
20
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 More
Number of DN packets before a COS is seen
Testing—10 ms RPI, 2 ms poll rate, 5 m range
3 MHz spread spectrum frequency across the 2.4 GHz bandwidth, generated at the base station
90
80
70
60
50
40
30
20
10
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 More
Number of DN packets before a COS is seen

(d)

FIGURE 23.14 (continued)

hardwired system with a "Packet Sniffer" node capability providing for real-time monitoring and time-stamping of information for performance evaluation. The experiment consists of DeviceNet requests sent out over the Bluetooth network to change the value (i.e., "flip") of a bit on the I/O block. When a packet is returned indicating that the bit value has been changed (i.e., a "change-of-state" is observed), a request is then placed to flip it back, etc. The packet interval of DeviceNet transmissions and poll rate of the system are set to be representative of typical DeviceNet operations and to ensure that the packet interval is significantly shorter than the response time of the Bluetooth system. Thus, a practical performance metric of "number of DeviceNet packets between observed COS occurrences" can be used. Figure 23.14b shows a baseline result with minimal noise interference and ideal distance between the two Bluetooth nodes; note the tight and bounded distribution around these packets indicating relatively high determinism. Figure 23.14c shows the same system, but with the distance increased to 10 m and the antennae of the two nodes oriented to be perpendicular to each other. The inset reveals that, while average performance does not degrade significantly, there is a significant loss of determinism due to the expanded tail of the distribution. Figure 23.13d illustrates that the same problem can occur when noise interference is introduced (note that the noise levels were purposely set to be below the threshold where the Bluetooth system would invoke frequency hopping to avoid noisy areas of the spectrum).

The results summarized in Figure 23.14 do not represent an exhaustive evaluation of Bluetooth, nor do they illustrate performance issues that are exclusive to Bluetooth. Rather the results illustrate one of the general problems with wireless that must be addressed if wireless is to be utilized in time-critical networked control and safety systems, namely, lack of determinism of the protocol in the face of external factors and the lack of focus on determinism in the design of the protocol as well as products and systems utilizing the protocol. The need for determinism in NCSs is explored further in Section 23.5.

23.4 NCS Characterization

Determination of an optimal NCS (which includes solutions for control, diagnostics, and/or safety applications) design for a particular manufacturing application requires not only knowing the theory and trade-offs of the NCS protocols being considered (i.e., a theoretical perspective), but also experimental data to augment the theoretical understanding of the NCS capabilities (i.e., experimental perspective), and an analytical study that allows for gauging the relative importance of performance metrics (e.g., end-to-end speed and jitter) to the specific application environment (i.e., analytical perspective). In this section, we explore these three perspectives in detail and show how they can be utilized collectively to provide a methodology for NCS design decision making [42].

23.4.1 Theoretical Perspective

The theoretical information perspective involves a general overview of network operation and the metrics that should be evaluated when assessing the best NCS solution for a particular system. The comparative evaluation of network technologies and implementations is primarily a study of trade-offs. Indeed, the move from point-to-point to networked systems itself is a study of trade-offs. One important aspect of the trade-off study is illustrated with the Lian curve, discussed in Section 23.2.4. This curve illustrates the impact of network congestion on NCS performance in a digital control environment. Finding and maintaining NCS operation in the sweet spot for optimized performance is a study of trade-offs that is a topic of ongoing research. It is important to note that the term "network" can be used very loosely in this analysis. For example, oftentimes the network congestion is actually observed in the devices themselves as they parse and encode data for end-to-end application communication [35]. The Lian curve phenomena can still be observed in these environments with the network definition extended to include the node processing.

A theoretical understanding of the network choices is a clear prerequisite to an NCS deployment decision. The understanding should extend beyond the network protocol performance to the entire system performance, and an important component of that understanding is the clear definition of performance metrics that are important to the particular application environment. An example of metrics that may be important in automotive manufacturing environment can be found in Ref. [1], which is the output of an industrial network performance workshop; in this case, the top five metrics were "node performance," "ease of use and diagnostics tools," "cost of security/technology," "capability for prioritization," and "time synchronization." The exercise of this workshop revealed a few important aspects of the metric selection process, which are important to the analytical application of these metrics in NCS decision making. The first is that many of the metrics such as ease-of-use or cost of complexity may be difficult to quantify, but these metrics must be incorporated into the quantified decision-making process (i.e., these metrics should not be avoided). Oftentimes, a qualitative metric can be broken down into quantitative sub-metrics (e.g., ease-of-use into average time between system crashes plus average time to diagnose a problem) so that it can be compared with other factors; the challenge is understanding the appropriate weighting for these quantifications. The second important aspect is that metrics can often overlap significantly. Care should be taken to break these metrics down into subcomponents so that a nonoverlapping set of subcomponents can be determined. This will avoid any "double counting" in the decision-making process.

23.4.2 Experimental Perspective

Usually, the theoretical knowledge that can be applied to NCS analysis must be augmented with experimental knowledge to complete the performance information base. This need for an experimental perspective is especially true when assessing the contribution of end node performance to the system performance analysis. Node performance should be evaluated in all systems because, as noted in Section 23.2, node performance has been shown to dominate over network performance in many manufacturing system environments. This is especially true in higher-level switched Ethernet systems, where the emphasis is on providing services such as security and self-typing on top of the data delivery mechanism [43,54]. These services provide overhead that contributes to network congestion; in smaller networks, where the node count does not number in the thousands, the degradation in individual node performance due to overhead factors such as these dominate. For example, Figure 23.15 illustrates that, in a particular two-node Ethernet network, node delay accounted for over 90% of the entire end-to-end delay [54]. Figure 23.5 (Section 23.2.3) gives an example of the impact of protocol overhead, in this case VPN to support security.

With respect to both network and node performance, specific network overhead factors that should be considered experimentally include self-typing or higher-level protocols for enabling control systems such as OPC (discussed in Section 23.3.5) and XML,* protocols for security such as VPN, and choices of Ethernet transport layer protocols such as TCP/IP vs. UDP. The impact of these factors is explored in detail in Ref. [54].

The wireless NCS application environment demands an additional level of experimental data to provide detailed information about the impact of the medium on NCS performance. In wired systems, the "wire" is generally considered to be a medium that can guarantee end-to-end delivery, provided that the protocol on top of it handles issues of congestion and performance (which is true for most industrial network protocols). There are number of factors in the various wireless manufacturing environments that can severely compromise wireless NCS performance. For example,

* eXtensible Markup Language (XML) is a self-typing language often used to support communications among dissimilar components on an industrial Ethernet network.

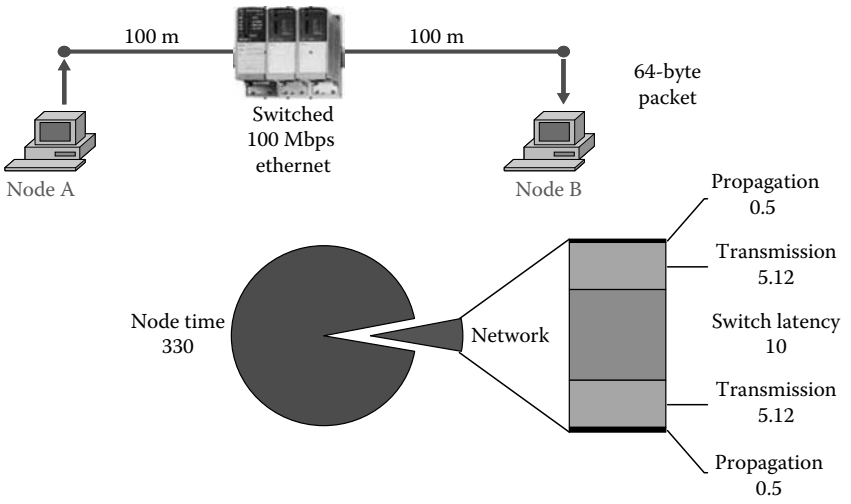


FIGURE 23.15 Illustration of node delay contribution to total delay in a typical two-node Ethernet network (pictured above). Time given in microseconds.

a mobile phone or a wire mesh used to strengthen a cement wall can lead to intermittent and difficult to diagnose NCS performance issues. Knowing the impact of these issues in advance helps optimize decisions as well as reduce discovery times for NCS performance problems detected during operation.

23.4.3 Analytical Perspective

While the theoretical and experimental analysis provides an excellent foundation for NCS evaluation, it does not provide a capability to combine the results for differential analysis with respect to the specifics of the application environment. A final analytical step is thus needed where the factors can be weighed and combined to yield a single answer with respect to the best decision. The weighted analysis can be achieved through a simple normalized weighting of factors of concern that are determined through either theoretical or experimental means [41]. The weighting equation takes the form of

$$WCost_{\text{total}} = \frac{W_{\text{H}}}{(W_{\text{H}} + W_{\text{E}} + W_{\text{P}})} Cost_{\text{H}} + \frac{W_{\text{E}}}{(W_{\text{H}} + W_{\text{E}} + W_{\text{P}})} Cost_{\text{E}} + \frac{W_{\text{P}}}{(W_{\text{H}} + W_{\text{E}} + W_{\text{P}})} Cost_{\text{P}} \tag{23.3}$$

where

$Cost_{\text{H}}$, $Cost_{\text{E}}$, and $Cost_{\text{P}}$ are the costs associated with hardware, engineering, and maintenance and performance

W_{H}, W_{E}, and W_{P} are the weights assigned, respectively, to these costs

Each cost factor is made up of a number of contributors, which are themselves weighted in a normalized manner as in

$$Cost_{\text{H}} = \frac{W_1}{\sum_{i=1}^{n} W_i} \cdot Cost_{\text{H1}} + \frac{W_2}{\sum_{i=1}^{n} W_i} \cdot Cost_{\text{H2}} + \cdots + \frac{W_n}{\sum_{i=1}^{n} W_i} \cdot Cost_{\text{H}n} \tag{23.4}$$

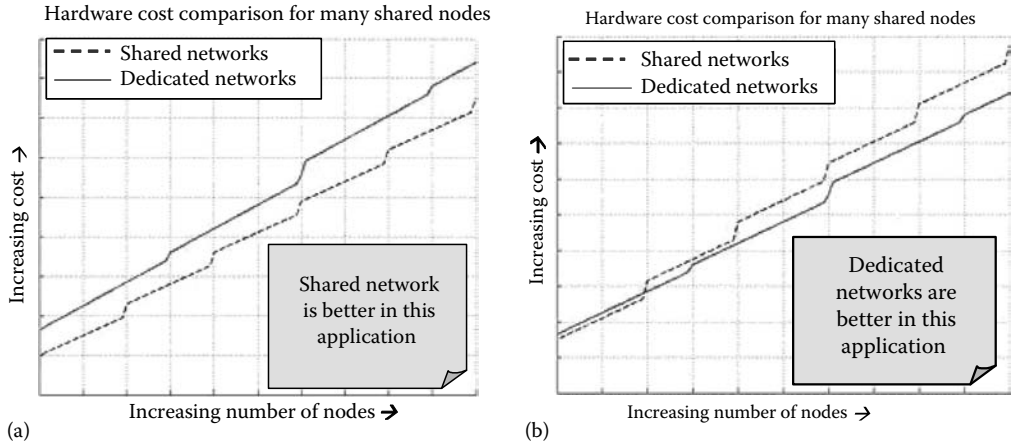


FIGURE 23.16 Illustration of the application of the weighted analytical perspective resulting in decisions of (a) shared or (b) dedicated networks for supporting control and safety functionality, respectively, depending on weights chosen.

The weights should be chosen to reflect the particular NCS application environment. Equation 23.3 is normalized so that factors can be incrementally added to the formulation as these factors are quantified during the course of understanding the NCS environment. Note that Equations 23.3 and 23.4 assume that the factors are independent; thus, as noted earlier, care should be taken to ensure that factors do not overlap.

Figure 23.16 illustrates this analysis applied to the decision of utilizing (separate) dedicated vs. shared networks to support control and safety NCS functionality. Depending on the prioritization of factors, the best solution can vary.

It is important to note that manufacturers tend to consistently overemphasize upfront costs such as installation and underemphasize recurring maintenance and performance costs when making network decisions. Utilizing the more formalized approach presented here, this "forgetting" of the recurring costs can be avoided [41].

23.5 Applications for Industrial Networks

In this section, we briefly describe current trends in the use of networks for distributed multilevel control, diagnostics, and safety. There is an enormous amount of data produced in a typical manufacturing system, as thousands of sensors record position, velocity, flow, temperature, and other variables hundreds of times every minute. In addition to this physical information, there are the specifications for the parts that must be produced, the orders for how many parts of each type are needed, and the maintenance schedules for each machine. Generally, the information content can be thought of as supporting a control, diagnostics, or safety function, or some combination of these. To support the aggregate of these functions in a manufacturing environment, separate networks are often employed, where each network is dedicated to one or more function types, such as control and diagnostics, with the network protocol chosen that best fits (i.e., balances) the QoS requirements of the function type(s). In this section, networks for these function types are explored, focusing on the QoS requirements that often govern the network protocol choice.

23.5.1 Networks for Control

Control signals can be divided into two categories: real-time and event-based. For real-time control, signals must be received within a specified amount of time for correct operation of the system. Examples of real-time signals include continuous feedback values (e.g., position, velocity, acceleration) for servo systems, temperature and flow in a process system, and limit switches and optical sensors in material flow applications (e.g., conveyors). To support real-time control, networks often must have a high level of determinism, i.e., they must be able to guarantee end-to-end communication of a signal within a specified amount of time. Further, QoS of NCSs can be very dependent upon the amount of jitter in the network; thus, for example, fixed determinism is usually preferred over bounded determinism.

Event-based control signals are used by the controller to make decisions, but do not have a time deadline. The system will wait until the signal is received (or a timeout is reached) and then the decision is made. An example of an event-based signal is the completion of a machining operating in a CNC; the part can stay in the machine without any harm to the system until a command is sent to the material handler to retrieve it.

In addition to dividing control signals by their time requirements, the data size that must be transmitted is important. Some control signals are a single bit (e.g., a limit switch) whereas others are very large (e.g., machine vision). Generally speaking, however, and especially with real-time control, data sizes on control networks tend to be relatively small and high levels of determinism are preferred.

Control networks in a factory are typically divided into multiple levels to correspond to the factory control distributed in a multitier hierarchical fashion. At the lowest level of networked control are device networks, which are usually characterized by smaller numbers of nodes (e.g., less than 64), communicating small data packets at high sample frequencies and with a higher level of determinism. An example of networked control at this level is servo control; here network delay and jitter requirements are very strict. Deterministic networks that support small data packet transmissions, such as CAN-based networks, are very common at this level. Although seemingly nonoptimal for this level of control, Ethernet is becoming more common, due to the desire to push Ethernet to all levels in the factory and the increasing determinism possible with switched Ethernet. Regardless of the network type, determinism and jitter capabilities for lower-level networked control are enhanced oftentimes by utilizing techniques that minimize the potential for jitter through network contention, such as MS operation, polling techniques, and deadbanding [52].

An intermediate level of network is the cell or subsystem, which includes SCADA. At this level, multiple controllers are connected to the network (instead of devices directly connected to the network). The controllers exchange both information and control signals, but as the cells or subsystems are typically physically decoupled, the timing requirements are not as strict as they are at the lowest levels, or nonexistent if event-driven control is enforced [49]. These networks are also used to download new part programs and updates to the lower-level controllers. TP and Ethernet-based networks are commonly used at this level, with ability to communicate larger amounts of data and support for network services generally taking precedence over strict determinism.

Networks at the factory or enterprise level coordinate multiple cells and link the factory-floor control infrastructure to the enterprise level systems (e.g., part ordering, supply chain integration, etc.) Large amounts of data travel over these networks, but the real-time requirements are usually nonexistent. Ethernet is the most popular choice here primarily because Internet support at this level is usually critical, and Ethernet also brings attractive features to this environment such as support for high data volumes, network services, availability of tools, capability for wide area distribution, and low cost.

Currently, wireless networks are rarely utilized for control. When they are utilized, it is often because a wired system is impossible or impractical (e.g., a stationary host controller for an autonomous guided vehicle [AGV]). Wireless systems are more applicable to discrete control; issues

with determinism of wireless communications in the face of interference (as explored in Section 23.4) usually result in the requirement that wireless time-critical control systems go through extensive and often costly verification prior to deployment.

23.5.2 Networks for Diagnostics

Diagnostic information that is sent over the network often consists of large amounts of data sent infrequently. For example, a tool monitoring system may capture spindle current at 1 kHz. The entire current trace would be sent to the diagnostic system after each part is cut (if the spindle current is used for real-time control, it could be sent over the network every 1 ms, but this would then be considered control data). The diagnostic system uses this information for higher-level control, such as to schedule a tool change or shut down a tool that is underperforming.

Diagnostics networks are thus usually set up to support large amounts of data with the emphasis on speed over determinism. Ethernet is the dominant network protocol in system diagnostics networks. As with control, diagnostics is often set up in a multitier hierarchical fashion, with different physical layer technology (e.g., wireless, broadband, and fiberoptic) utilized at different levels to support the data volume requirements. Also, a variety of data compression techniques, such as COS reporting and store and forwarding of diagnostic information on a process "run-by-run" basis, are often used in communicating diagnostic information up the layers of the network hierarchy [28,61]. Wireless is an ideal medium for diagnostics because determinism and high levels of data integrity are usually not required, and diagnostics systems can often take advantage of the flexibility afforded by the wireless media. As an example, a personal digital assistant hand-held system can be designed to collect diagnostics data wirelessly from a nearby machine and perform system health checks. Maintenance engineers thus could utilize a single portable unit as a diagnostics tool for all machines in a factory.

As noted in Section 23.1, diagnostics networks enable diagnostics of the networked system rather than the network itself (with the latter referred to as network diagnostics). Both types of diagnostics are commonly used in manufacturing systems. Many network protocols have built-in network diagnostics. For example, nodes that are configured to send data only when there is a COS may also send "heartbeat" messages every so often to indicate that they are still on the network.

23.5.3 Networks for Safety

One of the newest applications of networks in manufacturing is safety [2]. Traditionally, safety interlocks around manufacturing cells have been hardwired using ultrareliable safety relays to ensure that the robots and machines in the cell cannot run if the cell door is open or there is an operator inside the cell. This hardwiring is not easy to reconfigure and can be extremely difficult to troubleshoot if something goes wrong (e.g., a loose connection). Safety networks allow the safety systems to be reconfigurable and significantly improve the ability to troubleshoot. They also allow safety functions to be more easily coordinated across multiple components, e.g., shutting down all machines in a cell at the same time, and also coordinating "soft shutdown" where appropriate (safe algorithms for gradual shutdown of systems without damage to systems and/or scrapping of product). Further, safety network systems often provide better protection against operators bypassing the safety interlocks, and thus making the overall system safer.

Safety networks have the strongest determinism and jitter requirements of all network function types. Safety functions must be guaranteed within a defined time; thus, the network must provide that level of determinism. Further, the network must have a deterministic heartbeat-like capability; if network connectivity fails for any reason, the system must revert to a safe state within the guaranteed defined time; this is often referred to as "fail-to-safe." CAN-based networks are popular

candidates for networked safety because of their high levels of determinism and the network self-diagnostic mechanisms they can utilize to determine the node and network health. However, it is important to note that most network protocols, in and of themselves, are adequate to the task of supporting safety networking. This is because safety systems must support a fail-to-safe capability if a communication is not received within a predetermined amount of time. In other words, safety systems must be robust to the possibility of network failure of any network. This is often accomplished by adding functionality at higher levels (e.g., application) of the communication stack to guarantee proper safety functionality [16]. Thus, Ethernet and even wireless safety systems are possible. The key is understanding if the benefit of a more flexible or commonplace network technology, such as wireless or Ethernet, is outweighed by the reduced guaranteed response time of the safety system in the increased occurrence of fail-to-safe events due to network delays.

23.5.4 Multilevel Factory Networking Example: Reconfigurable Factory Testbed

The Reconfigurable Factory Testbed (RFT) at the University of Michigan is a comprehensive platform that enables research, development, education, validation, and transfer of reconfigurable manufacturing system concepts [45]. It consists of both real and virtual machines controlled over a communication network and coordinated through a unified software architecture. The RFT is conceived to be extensible to allow the modular incorporation and integration of additional components (hardware and/or software, real and/or virtual). The hardware components of the RFT include a serial–parallel manufacturing line consisting of two machining cells connected by a conveyor, a suite of communication and control networks, an AGV, and an RFID system. The software components of the RFT include a virtual factory simulation, an open software integration platform and data warehouse, an infrastructure of Web-based HMIs, and a computerized maintenance management system (CMMS). A schematic of the RFT is shown in Figure 23.17.

The network shown in Figure 23.17 represents a multitier-networked control, diagnostic, and safety network infrastructure that exists on the RFT. The serial–parallel line component of the RFT is the primary component currently being utilized to explore manufacturing networks. With respect to control networks, cell 1 has a DeviceNet network to connect the machines and robot controllers (including the robot gripper and the clamps in the machines); cell 2 uses PROFIBUS for the same purpose. The conveyor system (pallet stops, pallet sensors, motor, controller) was originally outfitted to communicate via a second DeviceNet network; recently this has been converted to an 802.11 wireless-networked system. The cell-level controllers (including the conveyor controller) communicate with the system level controller (SLC) over Ethernet via OPC and support an event-based control paradigm. The SLC has a wireless network connection with the AGV. All of these control networks are shown in Figure 23.18.

The network infrastructure for collecting the diagnostic data on the RFT uses OPC. For example, for every part that is machined, the spindle current on the machine is sampled at 1 kHz. This time-dense data is directly sampled using LabVIEW,* and then stored in the database. Compressed diagnostics data focused on identifying specific features of the current trace is passed to higher levels in the diagnostics network.

Networks for safety are implemented in the serial–parallel line utilizing the SafetyBUS p protocol, as shown in Figure 23.19. As with the control and diagnostics system, the implementation is multitier, corresponding to levels of safety control. Specifically, safety networks are implemented for each of the

* National Instruments, Austin, TX, USA.

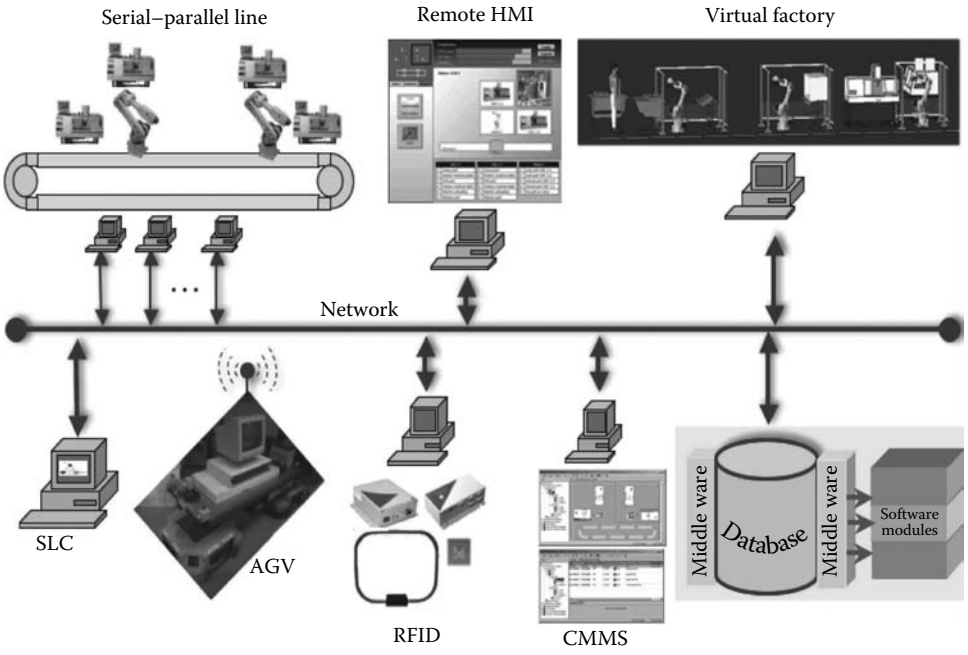


FIGURE 23.17 Reconfigurable factory testbed.

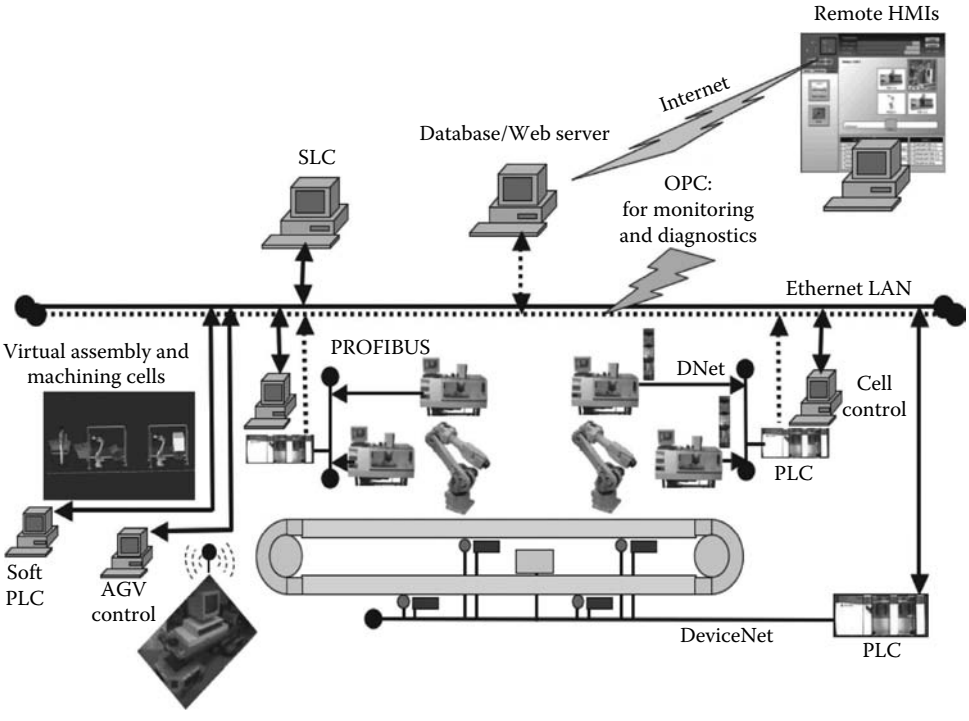


FIGURE 23.18 Networks on the RFT: control networks are indicated by solid lines and diagnostics networks are indicated by dashed lines.

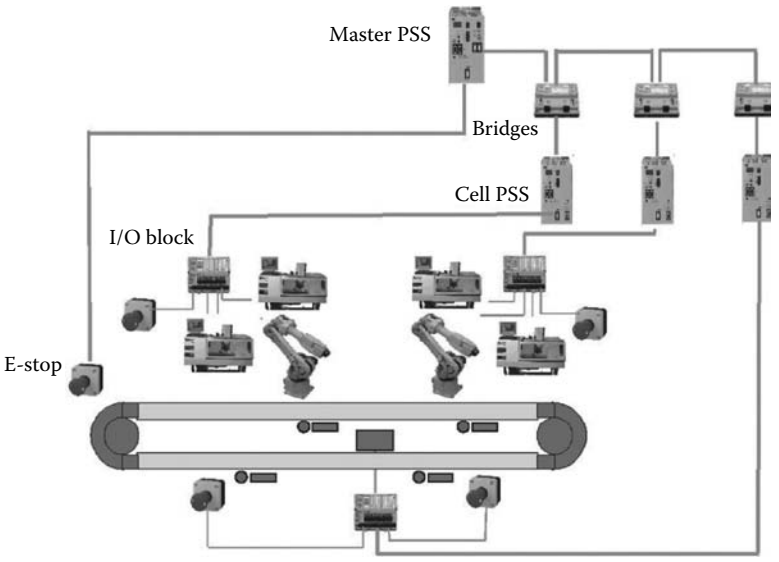


FIGURE 23.19 Safety network implementation on the RFT.

two cells as well as the conveyor. The safety network interlocks the emergency stops, robot cages, and machine enclosures with the controllers. These three cell-level networks are coordinated through a hierarchy to a high-level master safety network. This implementation not only allows for safety at each cell to be controlled individually, but also provides a capability for system-wide safe shutdown. Further, this implementation allows for multitier logic to be utilized for the implementation of safety algorithms.

As manufacturing networks research and development has increasingly shifted toward industrial Ethernet and wireless communications, so too has the focus in RFT networking deployment and analysis. For example, in providing an experimental capability to support NCS design evaluation (see Section 23.4), the wireless performance testbed shown in Figure 23.20 was developed in collaboration with USCAR, a collaboration of automotive manufacturers [68]. Here, a simple parallel point-to-point wireless environment is provided for three common industrial wireless protocols: IEEE 802.11, Bluetooth, and ZigBee. An Ethernet real-time packet "sniffer" is utilized to monitor traffic, thereby determining the performance of the wireless connection. The testbed is set up so that various wireless configurations and interference environments can be evaluated in a laboratory setting. The testbed can thus be utilized as both a design/evaluation as well as a trouble-shooting tool. With respect to the latter, wireless signal generation and analysis equipment are utilized with the testbed to effectively "record" wireless patterns in the field and then "play back" these patterns in the experimental environment for analysis and troubleshooting.

The RFT implementation of multitiered networks for control, diagnostics, and safety provides a rich research environment for exploration into industrial control networks. Specifically, topics that can be addressed include (1) coordination of control, diagnostics, and/or safety operation over one, two or three separate networks, (2) distributed control design and operation over a network, (3) distribution of control and diagnostics in a hierarchical networked system, (4) compression techniques for hierarchical diagnostics systems, (5) remote control safe operation, (6) hierarchical networked safety operation and soft shutdown, (7) heuristics for optimizing control/diagnostics/safety network operation, (8) network standards for manufacturing, as well as (9) best practices for network systems design and operation [33,34,45,53].

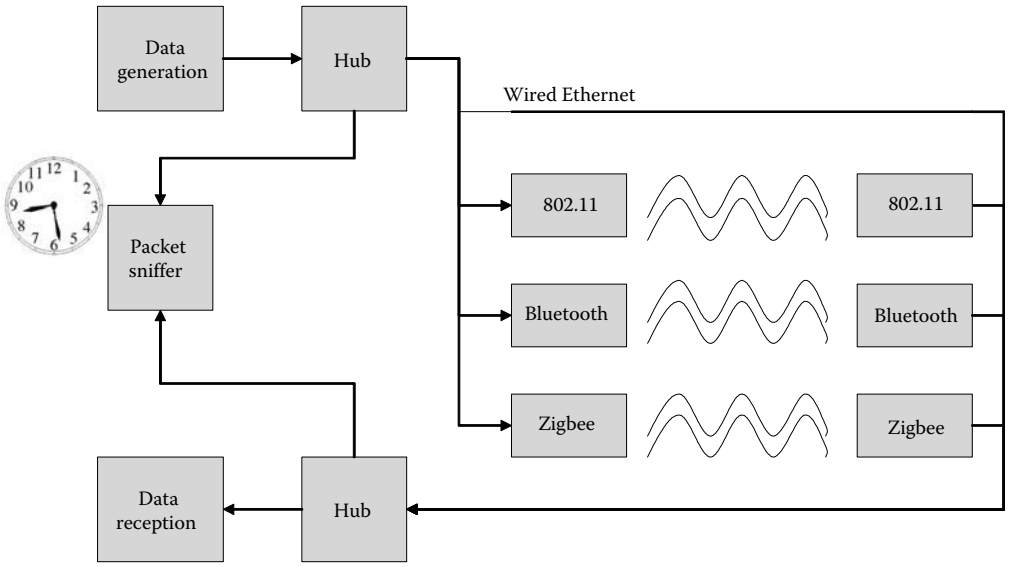


FIGURE 23.20 Experimental setup for determining and investigating the wireless NCS performance.

23.6 Future Trends

As the rapid move toward networks in industrial automation continues, the process has now matured to the point that a large portion of the migration is not just toward networks, but from one network technology to another (e.g., the move to Ethernet everywhere as a consolidation effort, or the move to wireless). The immediate advantages of reduced wiring and improved reliability have been accepted as fact in industry and are often significant enough by themselves (e.g., return-on-investment—ROI) to justify the move to networked solutions. Further, with the advent of more detailed ROI calculators such as that discussed in Section 23.4, other networking advantages can be quantified. It is expected that diagnostics network adoption will continue to lead the way in the overall network proliferation trend, followed by control and then safety networks, but the ordering is driven by the stricter QoS balancing requirements of the latter, not by any belief of higher ROI of the former. In fact, in gauging the criticality of control and safety with respect to diagnostics, it is conceivable that significantly higher ROI may be achievable in the migration to control and especially safety networking. However, even with diagnostics networks, the possibilities and benefits of e-Diagnostics and (with control) e-Manufacturing are only beginning to be explored.

Looking to the future, the most notable trend appearing in industry is the move to wireless networks at all levels [72]. This is not to say that wireless will replace wired in all applications, but rather the traditional barriers to wireless implementation (e.g., technology, security, reliability, etc.) will be lowered. This will allow the significant benefits of wireless to be realized, such as further reduction of the volume of wiring needed (although often power is still required), enabling of the placement of sensors in difficult locations, and enabling the placement of sensors on moving parts such as on tool tips that rotate at several thousand RPMs. Issues with the migration to wireless include interference between multiple wireless networks, security, and reliability and determinism of data transmission. The anticipated benefit in a number of domains (including many outside of manufacturing) is driving innovation that manufacturing in general should be able to leverage to address many of these issues. However, the determinism issue will be difficult to overcome because manufacturing is not a market driver in the typical application domains of most wireless network technologies.

Another noteworthy trend in manufacturing networked systems will be the integration of traditionally separate industrial systems, such as diagnostics and safety [59,60]. For example a diagnostics system can raise safety alarms when a process variable is outside the expected range of safe operations. Advancements in network partitioning allow the networked safety system operation to leverage networked diagnostics information into improved safety decision making, without degradation of overall safety network performance. This capability opens the door to a number of opportunities for improving safety, control, and diagnostics in manufacturing systems.

In summary, as the industry moves forward in networked communications, many among the dozens of protocols that have been developed for industrial networks over the last few decades will fall out of favor, but will not die overnight due to the large existing installed base and the long lifetime of manufacturing systems. In addition, new protocols may continue to emerge to address niches, where a unique QoS balance is needed. However, it is expected that Ethernet and wireless will continue to grab larger and larger shares of the industrial networks installed base, driven largely by lower cost through volume, the Internet, higher availability of solutions and tools for these network types (e.g., Web-enabled tools), and the unmatched flexibility of wireless. Indeed, it is not unreasonable to expect that, in the next decade, the next major milestone in industrial networking, namely, the "wireless factory," will be within reach, where diagnostics, control, and safety functions at multiple levels throughout the factory are enabled utilizing wireless technology.

Acknowledgments

The authors would like to acknowledge support from the NSF ERC for reconfigurable manufacturing systems (ERC-RMS) grant EEC95-92125. We would also like to thank the many students of the ERC-RMS who did much of the work on which much of this chapter is based, and Feng-Li Lian for his extensive research on NCSs.

References

1. K. Acton, M. Antolovic, N. Kalappa, J. Luntz, J. Moyne, and D. Tilbury, Practical metrics for evaluating network system performance. *UM-ERC/RMS Network Performance Workshop*, Ann Arbor, MI, April 2006. Available at http://erc.engin.umich.edu/publications
2. J. Alanen, M. Hietikko, and T. Malm. Safety of digital communications in machines. Technical Report VTT Tiedotteita—Research Notes 2265, VTT Technical Research Center of Finland, 2004. Available at http://www.vtt.fi/inf/pdf
3. P. Antsaklis and J. Baillieul (eds.). Special issue on networked control systems. *IEEE Transactions on Automatic Control*, 49(9):1421–1597, September 2004.
4. AS-I Standard, 2005. Available at http://www.as-interface.net
5. D. Bertsekas and R. Gallager. *Data Networks*, 2nd edn. Prentice-Hall, Englewood Cliffs, NJ, 1992.
6. Bluetooth Special Interest Group (SIG). Available at https://www.bluetooth.org/
7. D. Caro. *Wireless Networks for Industrial Automation*, 2nd edn. The Instrumentation, Systems and Automation Society, NC, USA, 2005.
8. B. J. Casey. Implementing Ethernet in the industrial environment. In *Proceedings of IEEE Industry Applications Society Annual Meeting*, vol. 2, pp. 1469–1477, Seattle, WA, October 1990.
9. J.-D. Decotignie. Ethernet-based real-time and industrial communications. *Proceedings of the IEEE*, 93(6):1102–1118, June 2005.
10. DeviceNet specifications, 1997.
11. L. Dugyard and E. I. Verriest. *Stability and Control of Time-Delay Systems*. Springer, New York, 1998.
12. A. Duschau-Wicke. Wireless monitoring and integration of control networks using OPC. Technical Report, NSF Engineering Research Center for Reconfigurable Manufacturing Systems, University of Michigan, 2004. Studienarbeit report for Technische Universit at Kaiserslautern.

13. D. Dzung, M. Naedele, T. P. Von Hoff, and M. Crevatin. Security for industrial communication systems. *Proceedings of the IEEE*, 93(6):1152–1177, June 2005.
14. M.-Y. Chow (ed.). Special section on distributed network-based control systems and applications. *IEEE Transactions on Industrial Electronics*, 51(6):1126–1279, December 2004.
15. J. Eidson and W. Cole. Ethernet rules closed-loop system. *InTech*, 36:39–42, June 1998.
16. Railway applications—Communication, signalling and processing systems. Part 1: Safety-related communication in closed transmission systems, 2001. Irish Standard EN 50159–1.
17. M. Felser. Real-time Ethernet—Industry prospective. *Proceedings of the IEEE*, 93(6):1118–1129, June 2005.
18. M. Felser and T. Sauter. The fieldbus war: History or short break between battles? In *Proceedings of the IEEE International Workshop on Factory Communication Systems* (*WFCS*), pp. 73–80, Västeras, Sweden, August 28, 2002.
19. M. Felser and T. Sauter. Standardization of industrial Ethernet—The next battlefield? In *Proceedings of the IEEE International Workshop on Factory Communication Systems* (*WFCS*), pp. 413–421, Vienna, Austria, September 2004.
20. M. Fondl. Network diagnostics for industrial Ethernet. *The Industrial Ethernet Book*, September 16, 2003. Available at http://ethernet.industrial-networking.com
21. G. F. Franklin, J. D. Powell, and A. Emani-Naeini. *Feedback Control of Dynamic Systems*, 3rd edn. Addison-Wesley, Reading, MA, 1994.
22. G. F. Franklin, J. D. Powell, and M. L. Workman. *Digital Control of Dynamic Systems*, 3rd edn. Addison-Wesley, Reading, MA, 1998.
23. Grid Connect. The Grid Connect Fieldbus comparison chart. Available at http://www.synergetic.com/compare.htm
24. D. W. Holley. Understanding and using OPC for maintenance and reliability applications. *IEE Computing and Control Engineering*, 15(1):28–31, February/March 2004.
25. IEC standard redefines safety systems. *InTech*, 50(7):25–26, July 2003.
26. IEEE. 1588: Standard for a precision clock synchronization protocol for networked measurement and control systems, 2002. Available at http://ieee1588.nist.gov
27. American National Standards Institute. *OSI Basic Reference Model*. ANSI, New York, 1984. ISO/7498.
28. International SEMATECH. *Proceedings of the International SEMATECH e-Manufacturing/e-Diagnostics Workshop*, April 2004. Available at http://ismi.sematech.org/emanufacturing/meetings/20040419/index.htm
29. H. Kaghazchi and D. Heffernan. Development of a gateway to PROFIBUS for remote diagnostics. In *PROFIBUS International Conference*, Warwickshire, U.K., 2004. Available online at http://www.ul.ie/~arc/techreport.html
30. S. A. Koubias and G. D. Papadopoulos. Modern fieldbus communication architectures for real-time industrial applications. *Computers in Industry*, 26:243–252, August 1995.
31. L. Larsson. Fourteen industrial Ethernet solutions under the spotlight. *The Industrial Ethernet Book*, (37): March 2007. Available at http://ethernet.industrial-networking.com
32. K. C. Lee and S. Lee. Performance evaluation of switched Ethernet for networked control systems. In *Proceedings of IEEE Conference of the Industrial Electronics Society*, 4:3170–3175, November 2002.
33. F. -L. Lian, J. M. Moyne, and D. M. Tilbury. Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet. *IEEE Control Systems Magazine*, 21(1):66–83, February 29, 2001.
34. F. -L. Lian, J. M. Moyne, D. M. Tilbury, and P. Otanez. A software toolkit for design and optimization of sensor bus systems in semiconductor manufacturing systems. In *AEC/APC Symposium XIII Proceedings*, Banff, Canada, October 2001.
35. F. -L. Lian, J. R. Moyne, and D. M. Tilbury. Network design consideration for distributed control systems. *IEEE Transactions on Control Systems Technology*, 10(2):297–307, March 2002.
36. P. S. Marshall. A comprehensive guide to industrial networks: Part 1. *Sensors Magazine*, 18(6): June 2001.

37. P. Marti, J. Yepez, M. Velasco, R. Villa, and J. M. Fuertes. Managing quality-of-control in network-based control systems by controller and message scheduling co-design. *IEEE Transactions on Industrial Electronics*, 51(6): December 2004.
38. G. A. Mintchell. OPC integrates the factory floor. *Control Engineering*, 48(1):39, January 2001.
39. J. Montague. Safety networks up and running. *Control Engineering*, 51(12): December 2004. Available at http://www.controleng.com/article/CA484725.html
40. J. Montague. Networks busting out all over. *Control Engineering*, 52(3): March 2005. Available at http://www.controleng.com/article/CA509788.html
41. J. Moyne, B. Triden, A. Thomas, K. Schroeder, and D. Tilbury. Cost function and tradeoff analysis of dedicated vs. shared networks for safety and control systems, *ATP International Journal*, 4(2):22–31, September 2006.
42. J. Moyne and D. Tilbury. Determining network control solution designs for manufacturing systems, In *Proceedings of 9th Biennial Conference on Engineering Systems Design and Analysis* (*ESDA'08*), Haifa, Israel, July 2008.
43. J. Moyne and D. Tilbury, Performance metrics for industrial Ethernet. *The Industrial Ethernet Book*, 38: April 2007. Available at http://ethernet.industrial-networking.com
44. J. Moyne and D. Tilbury, The emergence of industrial control networks for manufacturing control, diagnostics and safety data. *IEEE Proceedings, Special Issue on the Emerging Technology of Network Control Systems*, 95(1):29–47, January 2007.
45. J. Moyne, J. Korsakas, and D. M. Tilbury. Reconfigurable factory testbed (RFT): A distributed testbed for reconfigurable manufacturing systems. In *Proceedings of the Japan–U.S.A. Symposium on Flexible Automation*, Denver, CO, July 2004. American Society of Mechanical Engineers (ASME).
46. J. Moyne and F. Lian. Design considerations for a sensor bus system in semiconductor manufacturing. In *International SEMATECH AEC/APC Workshop XII*, September 2000.
47. J. Moyne, N. Najafi, D. Judd, and A. Stock. Analysis of sensor/actuator bus interoperability standard alternatives for semiconductor manufacturing. In *Sensors Expo Conference Proceedings*, Cleveland, OH, September 1994.
48. J. Moyne, P. Otanez, J. Parrott, D. Tilbury, and J. Korsakas. Capabilities and limitations of using Ethernet-based networking technologies in APC and e-Diagnostics applications. In *SEMATECH AEC/APC Symposium XIV Proceedings*, Snowbird, UT, September 2002.
49. J. Moyne, D. Tilbury, and H. Wijaya. An event-driven resource-based approach to high-level reconfigurable logic control and its application to a reconfigurable factory testbed. In *Proceedings of the CIRP International Conference on Reconfigurable Manufacturing Systems*, Ann Arbor, MI, 2005.
50. http://www.opcfoundation.org
51. OPC Foundation. OPC DA 3.00 Specification, March 2003. Available at http://www.opcfoundation.org, 30.
52. P. G. Otanez, J. R. Moyne, and D. M. Tilbury. Using deadbands to reduce communication in networked control systems. In *Proceedings of the American Control Conference*, pp. 3015–3020, Anchorage, Alaska, May 2002.
53. P. G. Otanez, J. T. Parrott, J. R. Moyne, and D. M. Tilbury. The implications of Ethernet as a control network. In *Proceedings of the Global Powertrain Congress*, Ann Arbor, MI, September 2002.
54. J. T. Parrott, J. R. Moyne, and D. M. Tilbury. Experimental determination of network quality of service in Ethernet: UDP, OPC, and VPN. In *Proceedings of the American Control Conference*, Minneapolis, 2006.
55. G. Paula. Java catches on for manufacturing. *Mechanical Engineering*, 119(12):80–82, December 1997.
56. PROFIBUS Standard. IEC 61158 type 3 and IEC 61784. Available at http://www.profibus.com
57. K. K. Ramakrishnan and H. Yang. The Ethernet capture effect: Analysis and solution. In *Proceedings of the 19th Conference on Local Computer Networks*, pp. 228–240, Minneapolis, MN, October 1994.
58. J.-P. Richard. Time-delay systems: An overview of some recent advances and open problems. *Automatica*, 39(10):1667–1694, 2003.

59. K. Schroeder, A. Khan, J. Moyne, and D. Tilbury. An application of the direct integration of industrial safety and diagnostics systems. In *Proceedings of the 2008 International Manufacturing Science and Engineering Conference* (*MSEC2008*), Evanston, IL, October 2008.
60. C. Scott, G. Law, et al. Integrated diagnostics in a process plant having a process control system and a safety system. U.S. Patent 6975966. United States of America, Fisher-Rosemount Systems, Inc. (Austin, TX). 2005.
61. A. Shah and A. Raman. Factory network analysis. In *Proceedings of the International SEMATECH e-Diagnostics and EEC Workshop*, July 2003. Available at http://ismi.sematech.org/ emanufacturing/meetings/20030718/index.htm
62. B. Shetler. OPC in manufacturing. *Manufacturing Engineering*, 130(6): June 2003.
63. P. Sink. Industrial Ethernet: The death knell of fieldbus? *Manufacturing Automation Magazine*, Canada, April 1999.
64. S. Soucek and T. Sauter. Quality of service concerns in IP-based control systems. *IEEE Transactions on Industrial Electronics*, 51:1249–1258, December 2004.
65. A. S. Tanenbaum. *Computer Networks*, 3rd edn. Prentice-Hall, Upper Saddle River, NJ, 1996.
66. J.-P. Thomesse. Fieldbus technology in industrial automation. *Proceedings of the IEEE*, 93(6):1073–1101, June 2005.
67. A. Treytl, T. Sauter, and C. Schwaiger. Security measures for industrial fieldbus systems—State of the art and solutions for IP-based approaches. In *Proceedings of the IEEE International Workshop on Factory Communication Systems* (*WFCS*), pp. 201–209, Vienna, Austria, September 31, 2004.
68. USCAR Web site: www.uscar.org
69. E. Vonnahme, S. Ruping, and U. Ruckert. Measurements in switched Ethernet networks used for automation systems. In *Proceedings of IEEE International Workshop on Factory Communication Systems*, pp. 231–238, September 2000.
70. K. Watanabe, E. Nobuyama, and A. Kojima. Recent advances in control of time delay systems—A tutorial review. In *Proceedings of the IEEE Conference on Decision and Control*, pp. 2083–2088, 1996.
71. J. D. Wheelis. Process control communications: Token bus, CSMA/CD, or token ring? *ISA Transactions*, 32(2):193–198, July 1993.
72. A. Willig, K. Matheus, and A. Wolisz. Wireless technology in industrial networks. *Proceedings of the IEEE*, 93(6):1130–1151, June 2005.
73. ZigBee Alliance. Available at http://www.zigbee.org

24

Wireless LAN Technology for the Factory Floor: Challenges and Approaches

24.1 Introduction **24**-1
24.2 Wireless Industrial Communications and Wireless Fieldbus: Challenges and Problems **24**-3
System Aspects • Real-Time Transmission over Error-Prone Channels • Integration of Wired and Wireless Stations/Hybrid Systems • Mobility Support • Security Aspects and Coexistence
24.3 Wireless LAN Technology and Wave Propagation **24**-6
Wireless LANs • Wave Propagation Effects
24.4 Physical Layer: Transmission Problems and Solution Approaches **24**-7
Effects on Transmission • Wireless Transmission Techniques
24.5 Problems and Solution Approaches on the MAC and Link Layer **24**-9
Problems for Wireless MAC Protocols • Methods for Combating Channel Errors and Channel Variation
24.6 Wireless Fieldbus Systems: State of the Art **24**-13
CAN • FIP/WorldFIP • PROFIBUS • Other Fieldbus Technologies
24.7 Wireless Ethernet/IEEE 802.11 ..... **24**-15
Brief Description of IEEE 802.11 • Real-Time Transmission over IEEE 802.11
24.8 Summary **24**-16
References **24**-16

Andreas Willig
Technical University of Berlin

24.1 Introduction

Wireless communication systems diffused into an ever-increasing number of application areas and achieved wide popularity. Wireless telephony and cellular systems are now an important part of our daily lives and wireless local area network (WLAN) technologies become more and more the primary way to access business and personal data. Two important benefits of wireless technology are key to this success: the need for cabling is greatly reduced and computers as well as users can be truly mobile. This saves costs and enables new applications. In factory plants, wireless technology can be used in several interesting ways [24, chap. 2]:

- WLAN technology can provide the communications services for distributed control applications involving mobile subsystems like autonomous transport vehicles, robots, or turntables.
- Implementation of distributed control systems in explosible areas or in the presence of aggressive chemicals.
- Frequent plant reconfiguration gets easier as fewer cables have to be remounted.
- Mobile plant diagnosis systems and wireless stations for programming and on-site configuration.

However, when adopting WLAN technologies for the factory floor some problems occur. The first problem is the tension between the hard reliability and timing requirements (hard real-time) pertaining to industrial applications on the one hand and the problem of wireless channels having time-variable and sometimes quite high error rates on the other. A second major source of problems is the desire to *integrate* wireless and wired stations into one single network (henceforth called a *hybrid system* or *hybrid network*). This integration calls for the design of interoperable protocols for the wired and wireless domains. Furthermore, using wireless technology imposes problems not anticipated in the original design of the (wired) fieldbus protocols: security problems, interference, mobility management, and so on.

In this chapter we survey some issues pertaining to the design and evaluation of protocols and architectures for (integrated) wireless industrial LANs and provide an overview of the state of the art. There is emphasis on aspects influencing the time and reliability behavior of wireless transmission. However, we discuss not only the problems but also present different solution approaches on the physical, medium access control (MAC), or data-link layer. These layers are key to the success of wireless fieldbus systems because they have important responsibilities in fulfilling timing and reliability requirements, and furthermore they are exposed most directly to the wireless link characteristics. In the second part of this chapter, we focus on technologies and the creation of hybrid systems. On the one hand, there are a number of existing fieldbus standards like controller area network (CAN), factory instrumentation protocol (FIP)/WorldFIP, or PROFIBUS. For these systems we discuss problems and approaches to create hybrid systems. On the other hand, one could start from existing wireless technologies and ask about their capabilities with respect to timeliness and reliability. The most widely deployed WLAN technology is currently the IEEE 802.11 WLAN standard and its suitability for industrial applications is discussed.

This chapter is structured as follows: In Section 24.2 important general considerations and problems of wireless industrial communications and wireless fieldbus systems are presented. In Section 24.3, we discuss some basic aspects of wireless LAN technology and wireless wave propagation. The transmission impairments resulting from certain wave propagation effects and some physical layer approaches to deal with them are presented in Section 24.4. Wireless wave propagation has also some interesting consequences on the operation of the MAC and data-link layer, these are discussed in Section 24.5. The following two sections, Section 24.6 and 24.7, take a more technology-oriented perspective. Specifically, in Section 24.6 we survey the state of the art regarding wireless industrial communication systems and wireless fieldbus systems. In Section 24.7 we present the important aspects of the IEEE 802.11 WLAN standard with respect to transmission of real-time data. Finally, in Section 24.8 we provide a brief summary.

The chapter restricts to protocol-related aspects of wireless transmission, other aspects like signal processing, analog and digital circuitry, and energy aspects are not considered. There are many introductory and advanced books on wireless networking, for example, [1,12,37,56,61,63,65,66]. Several separate topics in wireless communications are treated in [25]. Furthermore, this chapter is not intended to serve as introduction to fieldbus technologies, some background information can be found in [14,46].

24.2 Wireless Industrial Communications and Wireless Fieldbus: Challenges and Problems

In this section we survey some of the problem areas arising in wireless fieldbus systems.

24.2.1 System Aspects

First of all, wireless fieldbus systems will operate in similar environments as wired ones. Typically, a small to moderate number of stations are distributed over geographically small areas with no more than 100 m between any pair of stations [33]. Wired fieldbus systems offer bitrates ranging from hundreds of kilobits to (tens of) megabits per second, and wireless fieldbus systems should have comparable bitrates. The wireless transceivers have to meet electromagnetic compatibility requirements, meaning that they not only have to restrict their radiated power and frequencies, but also should be properly shielded from strong magnetic fields and electromagnetic noise emanated by strong motors, high voltage electrical discharges, and so on. This may pose a serious problem when off-the-shelf wireless transceivers are used (e.g., commercial IEEE 802.11 hardware), since these are typically designed for office environments and have no industrial-strength shielding.

Another problem is that many small fieldbus devices get their energy supply from the same wire as used for data transmission. If the cabling is to be removed from these devices, there is not only the problem of wireless data transmission but also the issue of wireless power transmission [31], which requires substantial effort.

For battery-driven devices the need to conserve energy arises. This has important consequences for the design of protocols [18,27] but is not discussed anymore in this chapter.

24.2.2 Real-Time Transmission over Error-Prone Channels

In industrial applications often *hard real-time* requirements play a key role. In accordance with [58] we assume the following important characteristics of hard real-time communications: (a) safety-critical messages must be transmitted reliably within an application-dependent deadline, (b) there should be support for message priorities to distinguish between important and unimportant messages, (c) messages with stringent timing constraints typically have a small size, and (d) both periodic and aperiodic/asynchronous traffic are present. The qualifier "hard" stems from the fact that losses or deadline misses of safety-critical packets can cost life or damage equipment. Both periodic and aperiodic messages in fieldbus systems can be subject to hard real-time constraints.

Wireless media tend to exhibit time-variable and sometimes high error rates, which creates a problem for fulfilling the hard real-time requirements. As an example, the measurements presented in [82] have shown that in a certain industrial environment for several seconds no packet gets through the channel. Therefore, seeking deterministic guarantees regarding timing and reliability is not appropriate. Instead, *stochastic guarantees* become important. An example formulation might be the percentage of safety-critical messages which can be transmitted reliably within a prespecified time-bound should be at least 99.x%. Of course, the error behavior limits the application areas of wireless industrial LANs—when deterministic guarantees in the range of 10–100 ms are essential, wireless transmission is ruled out (at least at the current state of art). However, if occasional emergency stop conditions due to message loss or missing deadlines are tolerable, wireless technologies can offer their potential. The goal is to reduce the frequency of losses and deadline misses.

It depends on the communication model how transmission reliability can be implemented. In many fieldbus systems (e.g., PROFIBUS) packets are transmitted from a sender to an *explicitly addressed* receiver station without involving other stations. Reliability can be ensured by several

mechanisms, for example, retransmissions, packet duplications, or error-correcting codes. On the other hand, systems like FIP/WorldFIP [73] and CAN [35] implement the model of a *real-time database* where *data* is identified instead of stations. A piece of data has one *producer* and potentially many *consumers*. The producer broadcasts the data and all interested consumers copy the data packet into an internal buffer. This broadcast approach prohibits the use of acknowledgments and packet retransmissions, but error-correcting codes can be still used to increase transmission reliability. Often the data is transmitted periodically and (repeated) packet losses can be detected by comparing the known period and the time of the last arrival of a data packet. This "freshness" information can be used by the application to react properly.

24.2.3 Integration of Wired and Wireless Stations/Hybrid Systems

There are a huge number of existing and productive fieldbus installations and it is best if wireless stations can be integrated into them. Such a network with both wireless stations (stations with a wireless transceiver) and wired stations are called *hybrid systems*. The most important requirements for hybrid systems are

- *Transparency*: There should be no need to modify the protocol stack of wired stations.
- *Using specifically tailored protocols*: Most fieldbus systems are specified on the layers one (physical), two (MAC and link layer), and seven (application layer). The introduction of a wireless physical layer affects the behavior and performance of both the MAC and link layer. The existing protocols for wired fieldbus systems are not designed for a wireless environment and should be replaced by protocols specifically tailored for the wireless link. However, this comes at the cost of *protocol conversion* between wired and wireless protocols.
- *Portability of higher-layer software*: If the link layer interface is the same for both the wireless and wired protocol stacks, implementations of higher-layer protocols and application software can be used in the same way on both types of stations.

The different approaches to integrate wireless stations into wired fieldbus LANs can be classified according to the layer of the OSI reference model where the integration actually happens [13,81]. Almost all fieldbus systems are restricted to the physical, data-link, and application layers [14]. The classification is as follows:

- *Wireless cable-replacement approach*: All stations are wired stations and thus attached to a cable. A piece of cable can be replaced by a wireless link and special bridge-like devices translate the framing rules used on the wireless and wired medium, respectively. In this approach, no station is aware of the wireless link. A typical application scenario is the wireless interconnection of two fieldbus segments.
- *Wireless MAC-unaware bridging approach*: The network comprises of both wired and wireless stations, but integration happens solely at the physical layer. Again, a bridge-like device translates the framing rules between wired and wireless media. The wireless stations use merely an alternative physical layer (PHY), but the MAC and link layer protocols remain the same as for wired stations.
- *Wireless MAC-aware bridging approach*: The LAN comprises of both wired and wireless stations and integration happens at the MAC and data-link layer. There are two different MAC and link layer protocol stacks for wired and wireless stations, but both offer the same link layer interface. The wireless MAC and link layer protocols should be (a) specifically tailored to the wireless medium and (b) easily integrable with the wired MAC and link

layer protocols. An intelligent bridge-like device is responsible for both translating the different framing rules as well as interoperation of the different MAC protocols.

- *Wireless gateway approach*: In this approach integration happens at the application layer or even in the application itself. Entirely different protocols can be used on the different media types.
- Some mixture of these approaches.

Any of these approaches require special *coupling devices* at the media boundaries. For the wireless cable-replacement and the MAC-unaware bridging approach, these devices can be simple. The other approaches may require complex and stateful operations. Hence, the issues of failures and redundancy need to be addressed.

24.2.4 Mobility Support

The potential station mobility is one of the main attractions of wireless systems. We can assume that wireless fieldbus systems will be mostly infrastructure-based (meaning that there are base stations or access points [APs]). A *handover* must be performed when a mobile station moves from the range of one AP into the range of another AP. Typically, handover processes involve exchange of signaling packets between the mobile and the APs. Ideally, a station can fulfill timing and reliability requirements even during a handover. The applicability and performance of handover schemes depend on the maximum speed of a mobile station. In industrial applications, it is typically forklifts, robots, or moving plant subsystems which are mobile, and it is safe to assume that these devices move with speed of no more than 20 km/h [33].

A simple consequence of mobility is that stations may enter and leave a network at unforeseeable times. To support this, a protocol stack at minimum must offer functionalities to make a new station known to the network/the other stations and sometimes also address assignment is needed. On the other hand, fieldbus systems and their applications often are designed with the assumption that the network is set up once and not changed afterward. Consequently, some fieldbus systems do not support any dynamics in their set of stations. Consider, for example, the FIP/WorldFIP fieldbus [73]. This system belongs to the class of real-time database systems and the producers of data items (called process variables) are polled by a dedicated central station, the *bus arbiter*. The bus arbiter keeps a table of variable identifiers and traverses it cyclically. To include a new station into the system, the arbiters table has to be modified by a human operator. It is worth noting that the most widely used fieldbus systems do not offer any support for dynamic address assignment.

24.2.5 Security Aspects and Coexistence

Security played no important role in the initial design of the fieldbus standards. This was reasonable, because physical access to a wire is needed to eavesdrop or inject packets. However, the introduction of wireless media allows an attacker to eavesdrop packets at some distance, for example on the factories parking lot. Even worse, an attacker could generate interference on the operating frequency of a wireless fieldbus system and distort all transmissions (including the time-critical and important ones). An attacker might also try to inject malicious packets into the network, for example, false valve commands. Therefore, security measures (integrity, authentication, authorization) have to be added to wireless fieldbus systems [64].

Noise and interference is not only generated purposely by some attacker, but can also come from colocated wireless systems working in the same frequency band. As an example, both IEEE 802.11 and Bluetooth use the 2.4 GHz ISM band and create mutual interference. This coexistence problem is explored in [10].

24.3 Wireless LAN Technology and Wave Propagation

In this section we discuss some basic characteristics of WLANs technology and present some of the fundamental wave propagation effects. In Sections 24.4 and 24.5, we discuss physical layer and MAC/link layer approaches to overcome or at least relax some of the problems created by the propagation effects.

24.3.1 Wireless LANs

Wireless LANs are designed for packet-switched communications over short distances (up to a few hundred meters) and with moderate to high bitrates. As an example, the IEEE 802.11 WLAN standard offers bitrates between 1 and 54 Mbps [54,69]. Wireless LANs usually use either infrared or radio frequencies. In the latter case license-free bands like the 2.4 GHz ISM band (industrial, scientific, and medical band) are particularly attractive, since the only restriction in using this band is a transmit power limit. On the other hand, since anyone can use these bands, several systems have to coexist. Radio waves below 6 GHz propagate through walls and can be reflected on several types of surfaces, depending on both frequency and material. Thus with radio frequencies non-line-of-sight communications is possible. In contrast, systems based on infrared only allow for line-of-sight (LOS) communications over a short distance. An example is the IrDA system [79].

Wireless LANs can be roughly subdivided into ad hoc networks [71] and infrastructure-based networks. In the latter case some centralized facilities like APs or base stations are responsible for tasks like radio resource management, forwarding data to distant stations, mobility management, and so on. In general, stations cannot communicate without the help of the infrastructure. In ad hoc networks, there is no prescribed infrastructure and the stations have to organize network operation by themselves.

Infrastructure-based WLANs offer some advantages for industrial applications. Many industrial communication systems already have an asymmetric structure which can be naturally accommodated in infrastructure-based systems. The often used master–slave communication scheme serves as an example. Furthermore, the opportunity to offload certain protocol processing tasks to the infrastructure allows to keep mobile stations more simple and to make efficient centralized decisions.

As compared to other wireless technologies like cellular systems and cordless telephony, WLAN technologies seem to offer the best compromise between data rate, geographical coverage, and license-free/independent operation.

24.3.2 Wave Propagation Effects

In the wireless channel, waves propagate through the air, which is an unguided medium. The wireless channel characteristics are significantly different from those of guided media like cables and fibers and create unique challenges for communication protocols.

A transmitted waveform is subjected to phenomena like path loss, attenuation, reflection, diffraction, scattering, adjacent- and co-channel interference, thermal or man-made noise, and finally to imperfections in the transmitter and receiver circuitry [8,61].

The path loss characterizes the loss in signal power when increasing the distance between a transmitter T and a receiver R. In general, the mean received power level $E[P_{Rx}]$ can be represented as the product of the transmit power P_{Tx} and the mean path loss $E[\mathrm{PL}]$:

$$E[P_{Rx}] = P_{Tx} \cdot E[\mathrm{PL}]$$

A typical path-loss model for omnidirectional antennas is given by [61, chap. 4.9]:

$$E[\mathrm{PL}](d) = C \cdot \left(\frac{d}{d_0}\right)^n$$

where

$d \geq d_0$ is the distance between T and R
$E[\mathrm{PL}](d)$ is the mean path loss
d_0 is a reference distance which depends on the antenna technology
C is a technology- and frequency-dependent scaling factor
n is the so-called path-loss exponent

Typical values for n are in the range between 2 (free-space propagation) and 5 (shadowed urban cellular radio), see also [61, chap. 4.9].

Reflection occurs when a waveform impinges on a smooth surface with structures significantly larger than the wavelength. Not all signal energy is reflected, some energy is absorbed by the material. The mechanism of diffraction allows a wave to propagate into a shadowed region, provided that some sharp edge exists. Scattering is produced when a wavefront hits a rough surface having structures smaller than the wavelength; it leads to signal diffusion in many directions.

The most important types of interference are co-channel interference and adjacent-channel interference. In co-channel interference a signal transmitted from T to R on channel c_1 is distorted by a parallel transmission on the same channel. In case of adjacent channel interference the interferer I transmits on an adjacent channel c_2, but due to imperfect filters R captures frequency components from c_2. Alternatively, an interferer I transmitting on channel c_2 leaks some signal energy into channel c_1 due to nonperfect transmit circuitry (amplifier).

Noise can be thermal noise or man-made noise. Thermal noise is created in the channel or in transceiver circuitry and can be found in almost any communications channel. Man-made noise in industrial environments can have several sources, for example, remote controls, motors, or microwave ovens.

24.4 Physical Layer: Transmission Problems and Solution Approaches

The previously discussed wave propagation effects can lead to channel errors. In general, their impact depends on a multitude of factors, including frequency, modulation scheme, and the current propagation environment. The propagation environment is characterized by the distance between stations, interferers, the number of different paths and their respective loss, and more. These factors can change when a station or parts of the environment move. Consequently the transmission quality is time-variable.

24.4.1 Effects on Transmission

The notion of slow fading refers to significant variations in the mean path loss, as they occur due to significant changes in distance between transmitter T and receiver R or by moving beyond large obstacles. Slow fading phenomena usually occur on longer timescales, they often coincide with human activity like mobility. For short durations in the range of a few seconds, the channel can often be assumed to have constant path loss.

An immediate result of reflection, diffraction, and scattering is that multiple copies of a signal may travel on different paths from T to R. Since these paths usually have different lengths, the copies arrive

at different times (delay spread) and with different phase shifts at the receiver and overlap. This has two consequences:

- Overlapping signals can interfere constructively or destructively. Destructive interference may lead to up to 40 dB loss of received power. Such a situation is often called a deep fade.
- Delay spread leads to inter-symbol interference, since signals belonging to neighbored information symbols overlap at the receiver.

If the stations move relative to each other or to the environment, the number of paths and their phase shifts vary in time. This results in a fast fluctuating signal strength at the receiver (called fast fading or multipath fading). It is important to note that these fluctuations are much faster than those caused by slow fading. Fast fading happens on the scale of milliseconds whereas slow fading happens at scales of seconds or minutes. On the timescale of milliseconds, the mean signal strength is constant. If the delay spread is small relative to the duration of a channel symbol, the channel is called non-frequency-selective or flat, otherwise it is called frequency-selective.

These problems translate into bit errors or packet losses. Packet losses occur when the receiver fails to acquire bit synchronization [82]. In case of bit errors, synchronization is successfully acquired, but a number of channel symbols is decoded incorrectly. The bit error rate can, for example, be reduced by using forward error correction (FEC) techniques [11,45]. The statistical properties of bit errors and packet losses were investigated in a number of studies [16,52,82]. While the results are not immediately comparable, certain trends show up in almost every study:

- Both bit errors and packet losses are "bursty," they occur in clusters with error-free periods between the clusters. The distributions of the cluster lengths and the lengths of error-free periods often have a large coefficient of variation or even seem to be heavy tailed.
- Bit error rates depend on the modulation scheme, typically schemes with higher bitrates/symbol rates exhibit higher error rates.
- Wireless channel is much worse than wired channels, often bit error rates of 10^{-3} to 10^{-6} can be observed. Furthermore, the bit error rate can vary over several orders of magnitudes within minutes.

Some knowledge about error generation patterns and error statistics can be helpful in designing more robust protocols.

24.4.2 Wireless Transmission Techniques

A number of different transmission techniques have been developed to combat the impairments of the wireless channel and to increase the reliability of data transmission.

Many types of WLANs (including IEEE 802.11) rely on spread-spectrum techniques [26], where a narrowband information signal is spread to a wideband signal at the transmitter and despread back to a narrowband signal at the receiver. By using a wideband signal, the effects of narrowband noise or narrowband interference are reduced. The two most important spread-spectrum techniques are direct sequence spread-spectrum (DSSS) and frequency hopping spread spectrum (FHSS).

In DSSS systems an information bit is multiplied (XORed) with a finite bipolar chip sequence such that transmission takes place at the chip rate instead of the information bitrate. The chip rate is much higher than the information rate and consequently requires more bandwidth; accordingly, the duration of a chip is much smaller than the duration of a user symbol. The chip rate is chosen such that the average delay spread is larger than the chip duration, thus the channel is frequency-selective. Receivers can explore this in different ways. To explain the first one, let us assume that the receiver receives a signal S from an LOS path and a delayed signal S' from another path, such that the delay difference (called lag) between S and S' is more than the duration of a single chip. The chip sequences

are designed such that the autocorrelation between the sequence and a shifted version of it is low for all lags of more than one chip duration. If a coherent matched-filter receiver is synchronized with the direct signal S, the delayed signal S' appears as white noise and produces only a minor distortion. In the RAKE receiver approach, delayed signal copies are not treated as noise but as a useful source of information [65, section 10.4]. Put briefly, a RAKE receiver tries to acquire the direct signal and the strongest time-delayed copies and combines them coherently. However, RAKE receivers are much more complex than simple matched-filter DSSS receivers.

In FHSS, the available spectrum is divided into a number of subchannels. The transmitter hops through the subchannels according to a predetermined schedule, which is also known to the receiver. The advantage of this scheme is that a subchannel currently subject to transmission errors is used only for a short time before the transmitter hops to the next channel. The hopping frequency is an important parameter of FHSS systems, since high frequencies require fast and accurate synchronization. As an example, the FHSS version of IEEE 802.11 hops with 2.5 Hz and many packets can be transmitted before the next hop. In Bluetooth the hopping frequency is 1.6 kHz and at most one packet can be transmitted before the next hop. Packets are always transmitted without being interrupted by hopping.

Recently, there has been considerable interest in orthogonal frequency division multiplexing (OFDM) techniques [75]. OFDM is a multicarrier technique, where blocks of N different symbols are transmitted in parallel over a number of N subcarriers. Hence, a single symbol has an increased symbol duration $N \cdot \tau$ as compared to full-rate transmission with symbol duration τ. The symbol duration $N \cdot \tau$ is usually much larger than the delay spread of the channel, this way combatting intersymbol-interference and increasing channel quality. The IEEE 802.11a standard [54] as well as HIPERLAN/2 [20,21] use an OFDM physical layer.

24.5 Problems and Solution Approaches on the MAC and Link Layer

The MAC and the link layer are exposed most to the error behavior of wireless channels and should do most of the work needed to improve the channel quality. Specifically for hard real-time communications, the MAC layer is a key component: if the delays on the MAC layer are not bounded, the upper layers cannot compensate this. In general, the operation of the MAC protocol is largely influenced by the properties of the physical layer. Some of the unique problems of wireless media are discussed in this section. For a general discussion of MAC and link layer protocols we refer the reader to [15,28,40,68,74].

24.5.1 Problems for Wireless MAC Protocols

Several problems arise due to path loss in conjunction with a threshold property—wireless receivers require the signal to have a minimum strength to be recognized. For a given transmit power, this requirement translates into an upper bound on the distance between two stations wishing to communicate;* if the distance between two stations is larger, they cannot hear each others transmissions. For MAC protocols based on carrier sensing (carrier sense multiple access [CSMA]), this property creates the hidden terminal problem [70] and the exposed terminal problem.

The hidden terminal problem is sketched in Figure 24.1. Consider three stations A, B, and C with transmission radii as indicated by the circles. Stations A and C are in range of B, but A is not in the

* To complicate things, wireless links are not necessarily bidirectional: it may well happen that station A can hear station B but not vice versa.

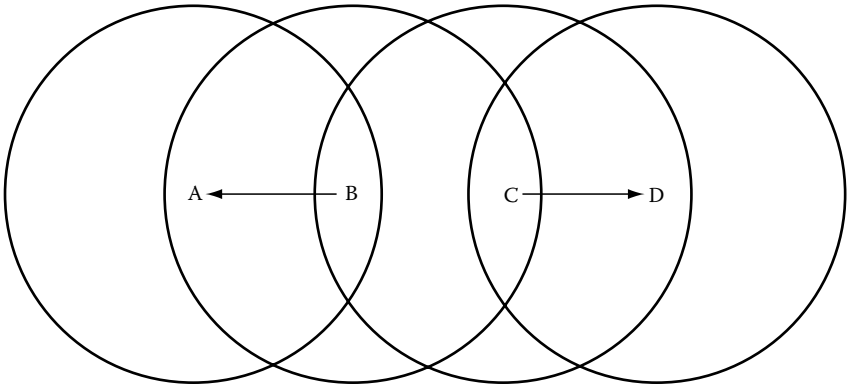


FIGURE 24.1 Hidden terminal scenario.

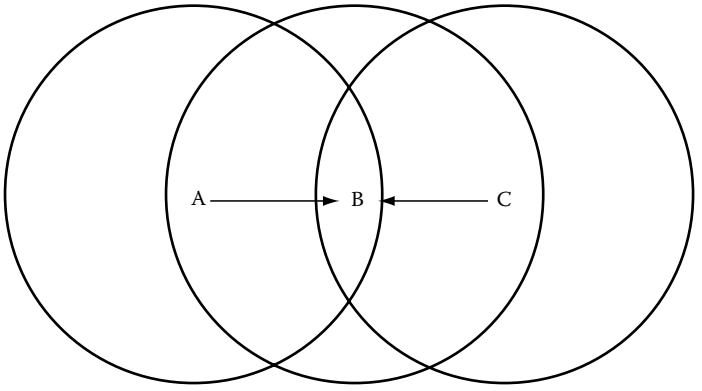


FIGURE 24.2 Exposed terminal scenario.

range of C and vice versa. If C starts to transmit to B, A cannot detect this by its carrier-sensing mechanism and considers the medium to be free. Consequently, A also starts packet transmission and a collision occurs at B.

The exposed terminal problem is also a result false prediction of the channel state at the receiver. An example scenario is shown in Figure 24.2. The four stations A, B, C, and D are placed such that the pairs A/B, B/C, and C/D can hear each other, all other combinations cannot. Consider the situation where B transmits to A, and one short moment later C wants to transmit to D. Station C performs carrier sensing and senses the medium busy due to B's transmission. Consequently, C postpones its transmission. However, C could safely transmit its packet to D without disturbing B's transmission to A. This leads to a loss of efficiency.

Two approaches to solve these problems are busy tone solutions [70] and the RTS/CTS protocol. In the busy tone solution two channels are assumed: a data channel and a separate control channel for the busy tone signals. The receiver of a packet transmits a busy tone signal on the control channel during packet reception. If a prospective transmitter wants to perform carrier sensing, it listens on the control channel instead of the data channel. If the control channel is free, the transmitter can start to transmit its packet on the data channel. This protocol solves the exposed terminal problem. The hidden terminal scenario is also solved except those rare cases where A and C start their transmissions simultaneously. However, if the busy tone is transmitted only when the receiver detects a valid packet header, the two colliding stations A and C can abort their transmissions quickly when they perceive the lack of a busy tone. The busy tone solution requires two channels and two transceivers.

The RTS/CTS protocol attacks the hidden terminal problem using only a single channel. Here we describe the variant used in the IEEE 802.11 WLAN, there are other ones. Consider the case that station A has a data packet for B. After A has obtained channel access it sends a short request-to-send (RTS) packet to B. This packet includes the time duration needed to finish the whole packet exchange sequence including the final acknowledgement. If B receives the RTS packet properly, it answers with a clear-to-send (CTS) packet, again including the time needed to finish the packet exchange sequence. Station A starts to transmit its data packet immediately after receiving the CTS packet. Any other station C, hearing the RTS and/or the CTS packet, defers its transmissions for the indicated time, this way not disturbing the ongoing packet transmission. It is a conservative choice to defer on any of the RTS or CTS packets, and in fact the exposed terminal problem still exists. One solution could be to let C defer only on reception of a CTS frame but to allow C a packet transmission if it hears an RTS without corresponding CTS frame.* The RTS/CTS protocol described here does not prevent collisions of RTS packets, it has significant overhead and it is still susceptible to subtle variants of the hidden terminal problem [60].

A significant problem of wireless transceivers is their inability to transmit and receive simultaneously on the same frequency band. Hence, a fast collision detection procedure similar to the CSMA/CD protocol of Ethernet is impossible to implement. Instead, collision detection has to resort to other mechanisms like the busy tone approach described above (rarely used) or the use of MAC layer acknowledgments (used frequently). Unfortunately, there are fieldbus systems relying on such a feature, for example, the CAN fieldbus [35] with its priority arbitration protocol. In this class of protocols each message is tagged with a priority value and this value is used to deterministically resolve collisions. In the CAN protocol, all stations are tightly time synchronized and the priority field is always at the start of a packet. All contending stations start packet transmission at the same time. Each contender transmits its priority field bit-by-bit and reads back the signal from the medium. If the medium state is the same as the transmitted bit, the station continues, otherwise the station gives up and waits for the next contention cycle. This protocol requires not only the ability to simultaneously listen and receive on the same channel, but channel shall also produce meaningful values from overlapping signals. Alternative implementations are sketched in Section 24.6.1.

Even receiver-based collision detection may not work reliably due to the near–far effect. Consider two stations A and B transmitting packets in parallel to a station C. For simplicity let us assume that both stations use the same transmit power. Station A is very close to C, whereas station B is far away but still in reach of C. Consequently, A's signal at C is much stronger than B's. In this case it may happen that C successfully decodes a packet sent by A despite B's parallel transmission. This situation is advantageous for the system throughput but disadvantageous for MAC protocols relying on collision detection or collision resolution.

24.5.2 Methods for Combating Channel Errors and Channel Variation

A challenging problem for real-time transmission is the error-prone and time-varying channel. There are many possible control knobs for improving the channel quality, for example, transmit power, bitrate/modulation, coding scheme/redundancy scheme, packet length, choice of retransmission scheme (automatic repeat request [ARQ]), postponing schemes and timing of (re-)transmissions, diversity schemes [57], and adaptation as a meta method [22]. In general, adaptation at the transmitter requires feedback from the receiver. This feedback can be created by using immediate acknowledgment packets after each data packet.

The variation of transmit power and of the bitrate/modulation scheme are both equivalent to varying the energy per bit, which in turn influences the bit error rate [59,61]. Roughly, higher

* Clearly, if C receives a distorted CTS packet it should defer.

transmit powers and slower bitrates/modulation schemes increase the probability of successful packet reception [32].

A common way to protect data bits against bit errors is to use redundancy. Example approaches are error detecting and correcting codes (also called FEC) [44] and the transmission of multiple copies of a packet [2]. The latter approach can also be classified as a time-diversity scheme [57]. It is beneficial for the overall throughput to control the amount of redundancy according to the current channel state such that none or only a little redundancy is added when the channel currently shows only few errors [9,16].

Another standard way to deal with transmission errors are retransmissions and suitable ARQ schemes. For channels with bursty errors, it is not clever to retransmit the same packet immediately on the same channel. Specifically, when the mean length of error bursts is of the same order or larger than the packet length, both the original packet and its immediate retransmission are likely to be hit by the same error burst. Hence, under these circumstances an immediate retransmission wastes time and energy.

The transmitter can *postpone* the retransmission for a while and possibly transmit packets to other stations/over other channels in the meantime. If the postponing delay is well chosen, the channel has left the error burst and the retransmission is successful. Indeed, it has been demonstrated in [4–6] that such an approach can reduce the number of wasted packets and increase the throughput significantly. But, how to choose the postponing delay? One option is to adopt some fixed value which could be based on measurements or on a priori knowledge about the channel. Another option is to send occasionally small probing packets [84] which the receiver has to acknowledge. If the transmitter captures such an acknowledgment, it assumes the channel to be back in good state and continues data transmission. For real-time systems, the postponing decision should not only consider the channel state but also the deadline of a packet. The authors of [17] describe a scheme which takes both the estimated channel state (for postponing decisions) and the packet deadline into account to select one coding scheme from a suite of available schemes.

Retransmissions do not necessarily need to use the same channel as the original packet. It is well known that wireless channels are spatially diverse: a signal transmitted by station A can be in a deep fade at geographical position p_1 and at the same time good enough to be properly received at another position p_2. This property is exploited by certain diversity techniques, for example, receiver diversity [61]. The receiver has two antennas and can pick the stronger/better of the two signals it reads from its antennas. If the spacing between the antennas is large enough,* the signals appear to be uncorrelated. The spatial diversity of wireless channels can also be explored at the protocol level. Assume that station A transmits a packet to station B. The channel from A to B is currently in a deep fade, but another station C successfully captures A's packet. If the channel from C to B is currently in a good state, the packet can be successfully transmitted over this channel. Therefore, station C helps A with its retransmission. This idea has been applied in [81] to the retransmission of data packets as well as to poll-packets in a polling-based MAC protocol.

In general, ARQ schemes can be integrated with FEC schemes into hybrid error control schemes [45]. Ideally, for industrial applications, deadlines should be taken into account when designing these schemes. In [3,72], retransmissions and FEC are combined with the concept of deadlines by increasing the coding strength with each retransmitted packet as the packet deadline approaches. This is called deadline-dependent coding. Another interesting hybrid error control technique is packet combining [30,39,78]. Put briefly, in these schemes the receiver tries to take advantage of the partially useful information contained in already received erroneous copies of a packet. For

* The two antennas should at minimum have a mutual distance of 40% of the wavelength [61, Chap. 5]. If the system works in the 2.4 GHz ISM band, this amounts to 5–6 cm.

example, if the receiver has received at least three erroneous copies of a packet, it can try to figure out the original packet by applying bit-by-bit majority voting. There are other packet combining techniques, for example equal-gain combining.

Sometimes the packet error probability (and therefore the need for retransmissions) can be reduced by proper tuning of packet sizes. Intuitively, it is clear that larger packets are more likely hit by errors than smaller ones. On the other hand, with smaller packets the fixed-size packet header becomes more dominant and leads to increased overhead. If the transmitter has estimates of current channel conditions it can choose the "appropriate" packet size giving the desired trade-off between reliability and efficiency [47].

24.6 Wireless Fieldbus Systems: State of the Art

Fieldbus systems are designed to deliver hard real-time services under harsh environmental conditions. A wireless fieldbus [13] should be designed to provide as stringent stochastic timing and reliability guarantees as possible over wireless links. However, in most of the literature surveyed in this section this issue is not addressed. Nonetheless, we discuss existing approaches for different popular fieldbus systems.

24.6.1 CAN

As already described in Section 24.5.1, the CAN system [35] uses a priority arbitration protocol on the MAC layer, which cannot be implemented directly on a wireless link. Some approaches have been developed to circumvent this; here we discuss a centralized and two distributed solutions [42].

The distributed WMAC protocol uses a CSMA/CA (carrier-sense-multiple-access with collision avoidance) scheme with priority-dependent backoffs. A station wishing to transmit a packet uses a carrier-sense mechanism to wait for the end of an ongoing packet transmission. After this, the station picks a backoff time depending on the priority value of the current packet. The station listens on the channel during the backoff time. If no other station starts transmission, the station assumes that it has the highest priority and starts transmitting its own packet. Otherwise, the station defers and starts over after the other packet has been finished. In another distributed scheme, the CAN message priority value is mapped onto the channel using an on–off keying scheme [41]. A station transmits a short burst if the current priority bit is a logical one, otherwise it switches into receive mode. If the station receives any signal it gives up, otherwise it continues with the next bit. The priority bits are considered from the most significant bit to the least significant bit. If the station is still contending after the last bit, it transmits the actual data packet. This approach requires tight synchronization and fast switching between transmit and receive modes of the radio transceiver, which is a problem for certain WLAN technologies.

The centralized RFMAC protocol leverages the fact that CAN belongs to the class of systems using the real-time database communication model. Data items are identified by unique identifiers. Similar to FIP/WorldFIP, all communications are controlled by a central station broadcasting the variable identifiers and causing the producers of the corresponding data items to transmit the data.

24.6.2 FIP/WorldFIP

The FIP/WorldFIP fieldbus uses a polling table to implement a real-time database [73]. To couple wired and wireless stations, in [49] a wireless-to-wired gateway is introduced, serving as central station for the wireless part. The wireless MAC protocol uses time division multiple access (TDMA), and each TDMA slot is used to transmit one data item (also called process variable).

In the OLCHFA project, a prototype system integrating wired and wireless FIP stations has been developed. This system worked in the 2.4 GHz ISM band using a DSSS physical layer [36]. The

available publications put emphasis on the management of configuration data and on distributed algorithms for clock synchronization. The MAC and data-link protocols of FIP were not modified. Since FIP broadcasts the values of process variables periodically, the protocol contains no retransmission scheme for the time-critical data. Instead, the OLCHFA approach is to enhance the FIP process variable model with the so-called time-critical variables, which provide freshness information to the applications. Applications can use this to handle cases of repeated losses.

24.6.3 PROFIBUS

The R-FIELDBUS project (www.rfieldbus.de) evaluated how IEEE 802.11 with DSSS can be used in a PROFIBUS fieldbus system and how such a system can be used for transmission of IP-based multimedia data [33,62]. Two different architectures have been proposed: the single logical ring and the multiple logical ring solution, discussed below. Both solutions run the (almost) unmodified PROFIBUS protocol. The PROFIBUS protocol uses token-passing on top of a broadcast medium. The token is passed between active stations along a logical ring and much of the protocols complexity deals with ring maintenance. The token itself is a small control frame.

In the single logical ring solution all wired and wireless stations are integrated into a single logical token-passing ring. The coupling devices between the wired and wireless domain simply forward all packets. This approach is easy to realize but subjects both data packets and control packets like the token-frame to the errors on wireless links. It is shown in [83] for the PROFIBUS and in [38] for the similar IEEE 802.4 Token Bus protocol that repeated losses of token-frames can create severe problems with the achievable real-time performance. Since there is only a single logical ring, the whole network is affected.

In contrast, in the multiple logical ring solution [23] wireless and wired stations are separated into several logical rings. These rings are coupled by intelligent devices called brouters (a merger from bridge and router). In this solution, transmission problems distort only one ring, the other logical rings remain operational. A second benefit of having multiple rings is traffic segmentation. If the segments are chosen carefully most of the traffic will be intra-segment and thus the overall traffic capacity can be increased. A drawback of the multiple logical ring solution, however, is that inter-segment traffic is not natively supported by the PROFIBUS protocol and extensions are required.

In [80,81] a system following the wireless MAC-aware bridging approach is proposed. On the wireless side specifically tailored polling-based protocols are used, whereas wired stations run the unmodified PROFIBUS protocol stack. The goal is to avoid token-passing on wireless segments. It is shown that for bursty channel errors the polling-based protocols achieve substantially better performance in terms of stochastic hard real-time behavior than the PROFIBUS token-passing protocol; for certain kinds of channels the 99% quantile of the delay needed to successfully transmit a high-priority packet is up to an order of magnitude smaller than for the PROFIBUS protocol. To integrate both protocols, the coupling device between wired and wireless media provides a virtual ring extension [80]. In this scheme the coupling device acts on the wired side on behalf of the wireless stations. For example, it creates token-frames and executes the ring maintenance mechanisms.

Finally, in [43] a scheme for integration of wireless nodes into a PROFIBUS-DP network (single master, many slaves, no token-passing) is described. An application layer gateway is integrated with a "virtual master" station. The virtual master acts as a proxy for the wireless stations, it polls them using standard IP and IEEE 802.11 distributed coordination function (DCF) protocols.

24.6.4 Other Fieldbus Technologies

For the IEC FieldBus [34] (which uses a centralized, polling-based access protocol for periodic data and a token-passing protocol for asynchronous data) in [7], an architecture which allows coupling

of several fieldbus segments using a wireless backbone based on IEEE 802.11 with point coordination function (PCF) is proposed.

In [50], it is investigated how the MAP/MMS application layer protocol can be enhanced with mobility. In the proposed system, the IEEE 802.11 WLAN with DCF is used; time-critical transmissions and channel errors are not considered. In [48], the same question was investigated with digital European cordless telephone (DECT) as underlying technology.

24.7 Wireless Ethernet/IEEE 802.11

Instead of developing WLAN technology for the factory floor from scratch, existing technologies might serve as a starting point. A good candidate is the IEEE 802.11 WLAN standard [53,54,69], since it is the most widely used WLAN technology. Some alternative systems are HIPERLAN [19,20], Bluetooth [29], and HomeRF [51].

24.7.1 Brief Description of IEEE 802.11

IEEE 802.11 belongs to the IEEE 802.x family of LAN standards. The standard describes architecture, services, and protocols for an Ethernet-like wireless LAN, using a CSMA/CA-based MAC protocol with enhancements for time-bounded services. The protocols run on top of several PHY's: an FHSS PHY, a DSSS PHY offering 1 and 2 Mbps [69], a 5.5 and 11 Mbps extension of the DSSS PHY [53], and an OFDM PHY with 54 Mbps [54].

The standard describes an ad hoc mode and an infrastructure-based mode. In the infrastructure mode all communications is relayed through fixed APs. An AP constitutes a service set in, and mobile stations have to associate with the closest AP. The APs are connected by a distribution system which allows to forward data packets between mobile stations in different cells. In the ad hoc mode, there are neither APs nor a distribution system, stations communicate in a peer-to-peer fashion. A detailed description of IEEE 802.11 can be found in [55].

The basic MAC protocol of 802.11 is called the DCF. It is a CSMA/CA protocol using the RTS/CTS scheme described in Section 24.5.1 and different inter-frame gaps to give control frames (e.g., acknowledgments, CTS frames) priority over data frames. However, data frames cannot be differentiated according to priorities. The IEEE 802.11 MAC provides a connectionless and semi-reliable best-effort service to its users by performing a bounded number of retransmissions. The user cannot specify any quality-of-service requirements for his packets, he can only choose between contention-based and contention-less transmission (see below). Furthermore, it is not possible to specify attributes like transmit power, modulation scheme, or the number of retransmissions on a per-packet basis. This control would be desirable for treating different packet types differently. As an example, one could transmit high-priority packets with high transmit power and low bitrate to increase their reliability.

The enhancement for time-bounded services is called PCF and works only in infrastructure mode. The PCF defines a superframe structure with variable—but maximum-length superframes. A superframe consists of a superframe header followed by a contention-free period (CFP) and a contention period (CP), both of variable length. During the CP all stations operate in DCF mode, including the APs. To initiate the start of the CFP, the AP (also called point coordinator, PC) has to acquire the wireless medium before it can transmit its beacon packet. Therefore, beacon transmissions and CFPs are not strictly periodic and isochronous services are not supported. The beacon indicates the length of the CFP and all stations receiving the beacon are forbidden to initiate transmissions during this time. Instead, they wait for being polled by the PC. If this happens, they can use the medium exclusively for transmission of a single packet. After the CFP ends the stations return to their usual DCF behavior and can initiate transmissions at their will.

The AP has a poll list of station addresses. The polling scheme itself is not fully specified. A station which desires to be polled has to signal this to the AP during the association process. The poll list membership ends upon disassociation or when the station reassociates itself without requesting contention-free service.

24.7.2 Real-Time Transmission over IEEE 802.11

The PCF is designed to provide time-bounded services. Many studies [67,76,77] confirm that indeed packets transmitted during the CFP receive substantially smaller delays than those transmitted during the CP, however, at the cost of substantial overhead. In [77] the authors show a scenario where eight voice calls of each having a data rate of 8 kbps require approximately 50% bandwidth of a 1 Mbps transmission medium (without channel errors).

When transmission has to be both timely and reliable despite channel errors, retransmissions are needed. When a time-critical packet transmitted during the CFP fails, the station can try the retransmission during the following CP or during the next CFP one superframe later (except the case where multiple entries in the polling list are allocated to the mobile and thus it receives multiple polls during the same CFP). Hence, retransmissions of important packets receive no priority in 802.11.

24.8 Summary

This chapter presented some problems and solution approaches to bring WLAN technology to the factory plant and to benefit from reduced cabling and mobility. The basic problem is the tension between the hard timing and reliability requirements of industrial applications on the one hand and the serious error rates and time-varying error behavior of wireless channels on the other hand. Many techniques have been developed to improve the reliability and timeliness behavior of lower layer wireless protocols, but up to now wireless fieldbus systems have not been deployed on a large scale as the problem of reliable transmission despite channel errors is not solved satisfactorily. It is not clear which combination of mechanisms and technologies has the potential to bound the number of deadline misses under realistic channel conditions.

It seems to be an open question whether just some more engineering is needed to make wireless transmission suitable for fulfilling hard real-time and reliability requirements, or whether there is really a limit of what can be achieved. Fortunately, wireless communications and WLAN technology is a very active field of research and development. New technologies are created and existing technologies are enhanced. As an example, the IEEE 802.11g and IEEE 802.11e working groups are working on delivering higher bitrates and better quality of service to users. It will be exciting to see how industrial applications can benefit from this.

References

1. L. Ahlin and J. Zander. *Principles of Wireless Communications*. Studentlitteratur, Lund, Sweden, 1998.
2. A. Annamalai and V. K. Bhargava. Analysis and optimization of adaptive multicopy transmission ARQ protocols for time-varying channels. *IEEE Transactions on Communications*, 46(10):1356–1368, 1998.
3. H. Bengtsson, E. Uhlemann, and P. -A. Wiberg. Protocol for wireless real-time systems. In *Proceedings of 11th Euromicro Conference on Real-Time Systems*, York, England, 1999.
4. P. Bhagwat, P. Bhattacharya, A. Krishna, and S. K. Tripathi. Using channel state dependent packet scheduling to improve TCP throughput over wireless LANs. *Wireless Networks*, 3(1):91–102, March 1997.
5. R. Cam and C. Leung. Multiplexed ARQ for time-varying channels—part I: System model and throughput analysis. *IEEE Transactions on Communications*, 46(1):41–51, January 1998.

6. R. Cam and C. Leung. Multiplexed ARQ for time-varying channels—part II: Postponed retransmission modification and numerical results. *IEEE Transactions on Communications*, 46(3):314–326, March 1998.
7. S. Cavalieri and D. Panno. On the integration of fieldbus traffic within IEEE 802.11 wireless LAN. In *Proceedings of 1997 IEEE International Workshop on Factory Communication Systems* (*WFCS'97*), Barcelona, Spain, 1997.
8. J. K. Cavers. *Mobile Channel Characteristics*. Kluwer Academic Publishers, Boston, Dordrecht, 2000.
9. R. Chen, K. C. Chua, B. T. Tan, and C. S. Ng. Adaptive error coding using channel prediction. *Wireless Networks*, 5(1):23–32, February 1999.
10. C. -F. Chiasserini and R. R. Rao. Coexistence mechanisms for interference mitigation in the 2.4-GHz ISM band. *IEEE Transactions on Wireless Communications*, 2(5):964–975, September 2003.
11. D. J. Costello, J. Hagenauer, H. Imai, and S. B. Wicker. Applications of error-control coding. *IEEE Transactions on Information Theory*, 44(6):2531–2560, October 1998.
12. K. David and T. Benkner. *Digitale Mobilfunksysteme*. Informationstechnik. B.G. Teubner, Stuttgart, 1996.
13. J. -D. Decotignie. Wireless fieldbusses—a survey of issues and solutions. In *Proceedings of 15th IFAC World Congress on Automatic Control* (*IFAC 2002*), Barcelona, Spain, 2002.
14. J. -D. Decotignie and P. Pleineveaux. A survey on industrial communication networks. *Annals of Telecommunications*, 48(9):435–448, 1993.
15. L. Dellaverson and W. Dellaverson. Distributed channel access on wireless ATM links. *IEEE Communications Magazine*, 35(11):110–113, November 1997.
16. D. A. Eckhardt and P. Steenkiste. A trace-based evaluation of adaptive error correction for a wireless local area network. *MONET—Mobile Networks and Applications*, 4:273–287, 1999.
17. M. Elaoud and P. Ramanathan. Adaptive use of error-correcting codes for real-time communication in wireless networks. In *Proceedings of IEEE INFOCOM 1998*, San Francisco, March 1998.
18. A. Ephremides. Energy concerns in wireless networks. *IEEE Wireless Communications*, 9(4):48–59, August 2002.
19. ETSI. *High Performance Radio Local Area Network* (*HIPERLAN*)—*Draft Standard*. ETSI, March 1996.
20. ETSI. *TR 101 683*, *HIPERLAN Type 2*: *System Overview*. ETSI, February 2000.
21. ETSI. *TS 101 475*, *BRAN*, *HIPERLAN Type 2*: *Physical* (*PHY*) *Layer*. ETSI, March 2000.
22. A. Farago, A. D. Myers, V. R. Syrotiuk, and G. V. Zaruba. Meta-MAC protocols: Automatic combination of MAC protocols to optimize performance for unknown conditions. *IEEE Journal on Selected Areas in Communications*, 18(9):1670–1681, September 2000.
23. L. Ferreira, M. Alves, and E. Tovar. Hybrid Wired/Wireless PROFIBUS Networks Supported by Bridges/Routers. In *Proceedings of 2002 IEEE Workshop on Factory Communication Systems*, *WFCS'2002*, pp. 193–202, Västeras, Sweden, 2002.
24. F. -P. Das. Verbundprojekt Drahtlose Feldbusse im Produktionsumfeld (Funbus)—Abschlußbericht. INTERBUS Club Deutschland e.V., Postf. 1108, 32817 Blomberg, Bestell-Nr: TNR 5121324, October 2000. http://www.softing.de/d/NEWS/Funbusbericht.pdf.
25. J. D. Gibson, editor. *The Communications Handbook*. CRC Press/IEEE Press, Boca Raton, Florida, 1996.
26. S. Glisic and B. Vucetic. *Spread Spectrum CDMA Systems for Wireless Communications*. Artech House, Boston, MA, 1997.
27. A. J. Goldsmith and S. B. Wicker. Design challenges for energy-constrained ad hoc wireless networks. *IEEE Wireless Communications*, 9(4):8–27, August 2002.
28. A. Chandra, V. Gummalla, and J. O. Limb. Wireless medium access control protocols. *IEEE Communications Surveys and Tutorials*, 3(2), 2–15, 2000. http://www.comsoc.org/pubs/surveys.
29. J. C. Haartsen. The Bluetooth Radio System. *IEEE Personal Communications*, 7(1):28–36, February 2000.
30. B. A. Harvey and S. B.Wicker. Packet combining systems based on the Viterbi decoder. *IEEE Transactions on Communications*, 42(2):1544–1557, February 1994.

31. J. Hirai, T. -W. Kim, and A. Kawamura. Practical study on wireless transmission of power and information for autonomous decentralized manufacturing system. *IEEE Transactions on Industrial Electronics*, 46:349–359, April 1999.
32. G. Holland, N. Vaidya, and P. Bahl. A Rate-Adaptive MAC Protocol for Wireless Networks. In *Proceedings of 7th Annual International Conference on Mobile Computing and Networking 2001* (*MobiCom*), Rome, Italy, July 2001.
33. J. Hähniche and L. Rauchhaupt. Radio communication in automation systems: the R-Fieldbus approach. In *Proceedings of 2000 IEEE International Workshop on Factory Communication Systems* (*WFCS 2000*), pp. 319–326, Porto, Portugal, 2000.
34. IEC—International Electrotechnical Commission. *IEC-1158-1*, *FieldBus Specification*, *Part 1*, *FieldBus Standard for Use in Industrial Control*: *Functional Requirements*, 1997.
35. International Organization for Standardization. *ISO Standard 11898—Road Vehicle—Interchange of Digital Information—Controller Area Network* (*CAN*) *for High-Speed Communication*. ISO—International Organization for Standardization, 1993.
36. I. Izikowitz and M. Solvie. Industrial needs for time-critical wireless communication & wireless data transmission and application layer support for time critical communication. In *Proceedings of Euro-Arch'93*, Munich, 1993. Springer Verlag, Berlin.
37. W. C. Jakes, editor. *Microwave Mobile Communications*. IEEE Press, Piscataway, NJ, 1993.
38. H. ju Moon, H. S. Park, S. C. Ahn, and W. H. Kwon. Performance degradation of the IEEE 802.4 token bus network in a noisy environment. *Computer Communications*, 21:547–557, 1998.
39. S. Kallel. Analysis of a type-II hybrid ARQ scheme with code combining. *IEEE Transactions on Communications*, 38(8):1133–1137, August 1990.
40. J. F. Kurose, M. Schwartz, and Y. Yemini. Multiple-access protocols and time-constrained communication. *ACM Computing Surveys*, 16:43–70, March 1984.
41. A. Kutlu, H. Ekiz, M. D. Baba, and E. T. Powner. Implementation of "Comb" based wireless access method for control area network. In *Proceedings of 11th International Symposium on Computer and Information Science*, pp. 565–573, Antalaya, Turkey, November 1996.
42. A. Kutlu, H. Ekiz, and E. T. Powner. Performance analysis of MAC protocols for wireless control area network. In *Proceedings of International Symposium on Parallel Architectures, Algorithms and Networks*, pp. 494–499, Beijing, China, June 1996.
43. K. C. Lee and S. Lee. Integrated network of PROFIBUS-DP and IEEE 802.11 wireless LAN with hard real-time requirement. In *Proceedings of IEEE 2001 International Symposium on Industrial Electronics*, Korea, 2001.
44. S. Lin and D. J. Costello. *Error Control Coding—Fundamentals and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1983.
45. H. Liu, H. Ma, M. E. Zarki, and S. Gupta. Error control schemes for networks: An overview. *MONET—Mobile Networks and Applications*, 2(2):167–182, 1997.
46. N. P. Mahalik, editor. *Fieldbus Technology—Industrial Network Standards for Real-Time Distributed Control*. Springer, Berlin, 2003.
47. E. Modiano. An adaptive algorithm for optimizing the packet size used in wireless ARQ protocols. *Wireless Networks*, 5:279–286, 1999.
48. P. Morel. Mobility in MAP networks using the DECT wireless protocols. In *Proceedings of 1995 IEEE Workshop on Factory Communication Systems, WFCS'95*, Leysin, Switzerland, 1995.
49. P. Morel and A. Croisier. A wireless gateway for fieldbus. In *Proceedings of 6th International Symposium on Personal, Indoor and Mobile Radio Communications* (*PIMRC 95*), Toronto, Canada, 1995.
50. P. Morel and J. -D. Decotignie. Integration of wireless mobile nodes in MAP/MMS. In *Proceedings of 13th IFAC Workshop on Distributed Computer Control Systems DCCS95*, Toulouse, France, 1995.
51. K. J. Negus, A. P. Stephens, and J. Lansford. HomeRF: Wireless networking for the connected home. *IEEE Personal Communications*, 7(1):20–27, February 2000.

52. G. T. Nguyen, R. H. Katz, B. Noble, and M. Satyanarayanan. A trace-based approach for modeling wireless channel behavior. In *Proceedings of the Winter Simulation Conference*, Coronado, CA, December 1996.
53. The Editors of IEEE 802.11. *IEEE Standard for Information Technology—Telecommunications and Information Exchange between Systems—Local and Metropolitan Networks—Specific Requirements, Part 11: Wireless LAN Medium Access Control* (*MAC*) *and Physical Layer* (*PHY*) *Specifications*: *Higher Speed Physical Layer* (*PHY*) *Extension in the 2.4 GHz Band*, 1999.
54. The Editors of IEEE 802.11. *IEEE Standard for Telecommunications and Information Exchange between Systems—LAN/MAN Specific Requirements, Part 11: Wireless Medium Access Control* (*MAC*) *and Physical Layer* (*PHY*) *Specifications*: *High Speed Physical Layer in the 5 GHz Band*, 1999.
55. B. O'Hara and A. Petrick. *IEEE 802.11 Handbook—A Designer's Companion*. IEEE Press, New York, 1999.
56. K. Pahlavan and A. H. Levesque. *Wireless Information Networks*. John Wiley & Sons, New York, 1995.
57. A. Paulraj. Diversity techniques. In J. D. Gibson, editor, *The Communications Handbook*, pp. 213–223. CRC Press/IEEE Press, Boca Raton, FL, 1996.
58. J. R. Pimentel. *Communication Networks for Manufacturing*. Prentice-Hall International, Upper Saddle River, NJ, 1990.
59. J. G. Proakis. *Digital Communications*, 3rd edition. McGraw-Hill, New York, 1995.
60. C. S. Raghavendra and S. Singh. Pamas—power aware multi-access protocol with signalling for ad hoc networks. *ACM Computer Communication Review*, 27:5–26, July 1998.
61. T. S. Rappaport. *Wireless Communications—Principles and Practice*. Prentice Hall, Upper Saddle River, NJ, 2002.
62. L. Rauchhaupt. System and device architecture of a radio-based fieldbus—the RFieldbus system. In *Proceedings of 4th IEEE Workshop on Factory Communication Systems 2002* (*WFCS 2002*), Västeras, Sweden, 2002.
63. A. Santamaria and F. J. Lopez-Hernandez, editors. *Wireless LAN—Standards and Applications. Mobile Communication Series*. Artech House, Boston, MA/London, 2001.
64. G. Schäfer. *Security in Fixed and Wireless Networks—An Introduction to Securing Data Communications*. John Wiley & Sons, Chichester, U.K., 2003.
65. W. Stallings. *Wireless Communications and Networks*. Prentice Hall, Upper Saddle River, NJ, 2001.
66. I. Stojmenovic, editor. *Handbook of Wireless Networks and Mobile Computing*. John Wiley & Sons, New York, 2002.
67. T. Suzuki and S. Tasaka. Performance evaluation of video transmission with the PCF of the IEEE 802.11 standard MAC protocol. *IEICE Transactions on Communications*, E83-B(9):2068–2076, September 2000.
68. A. S. Tanenbaum. *Computernetzwerke*, 3rd edition. Prentice-Hall, Munich, 1997.
69. The Editors of IEEE 802.11. *IEEE Standard for Wireless LAN Medium Access Control* (*MAC*) *and Physical Layer* (*PHY*) *Specifications*, November 1997.
70. F. A. Tobagi and L. Kleinrock. Packet switching in radio channels, Part II: The hidden terminal problem in CSMA and busy-tone solutions. *IEEE Transactions on Communications*, 23(12):1417–1433, 1975.
71. C. -K. Toh. *Ad Hoc Mobile Wireless Networks—Protocols and Systems*. Prentice Hall, Upper Saddle River, NJ, 2002.
72. E. Uhlemann, P. -A.Wiberg, T. M. Aulin, and L. K. Rasmussen. Deadline-dependent coding—a framework for wireless real-time communication. In *Proceedings of International Conference on Real-Time Computing Systems and Applications*, pp. 135–142, Cheju Island, South Korea, December 2000.
73. Union Technique de l'Electricit'e. *General Purpose Field Communication System*, EN 50170, Volume 3: *WorldFIP*, 1996.
74. H. R. van As. Media access techniques: The evolution towards terabit/s LANs and MANs. *Computer Networks and ISDN Systems*, 26:603–656, 1994.

75. R. van Nee and R. Prasad. *OFDM for Wireless Multimedia Communications*. Artech House, London, 2000.
76. M. Veeraraghavan, N. Cocker, and T. Moors. Support of voice services in IEEE 802.11 wireless LANs. In *Proceedings of IEEE INFOCOM 2001*, Anchorage, Alaska, April 2001.
77. M. A. Visser and M. El Zarki. Voice and data transmission over an 802.11 wireless network. In *Proceedings of IEEE Personal, Indoor and Mobile Radio Conference* (*PIMRC*) *95*, pp. 648–652, Toronto, Canada, September 1995.
78. X. Wang and M. T. Orchard. On reducing the rate of retransmission in time-varying channels. *IEEE Transactions on Communications*, 51(6):900–910, June 2003.
79. S. Williams. IrDA: Past, present and future. *IEEE Personal Communications*, 7(1):11–19, February 2000.
80. A. Willig. An architecture for wireless extension of PROFIBUS. In *Proceedings of IECON 03*, Roanoke, Virginia, 2003.
81. A. Willig. Polling-based MAC protocols for improving realtime performance in a wireless PROFIBUS. *IEEE Transactions on Industrial Electronics*, 50(4):806–817, August 2003.
82. A. Willig, M. Kubisch, C. Hoene, and A. Wolisz. Measurements of a wireless link in an industrial environment using an IEEE 802.11-compliant physical layer. *IEEE Transactions on Industrial Electronics*, 49(6):1265–1282, 2002.
83. A. Willig and A. Wolisz. Ring stability of the PROFIBUS token passing protocol over error prone links. *IEEE Transactions on Industrial Electronics*, 48(5):1025–1033, October 2001.
84. M. Zorzi and R. R. Rao. Error control and energy consumption in communications for nomadic computing. *IEEE Transactions on Computers*, 46:279–289, March 1997.

25

Wireless Local and Wireless Personal Area Network Communication in Industrial Environments

25.1 Introduction 25-1
25.2 WLAN, WPAN, WSN, Cellular and Ad Hoc Networks 25-3
25.3 Bluetooth Technology 25-4
Technical Background • Performance
25.4 IEEE 802.11 25-9
Technical Background • Performance
25.5 IEEE 802.15.4/ZigBee 25-15
Technical Background • Performance
25.6 Coexistence of WPAN and WLAN 25-17
Separation in the Frequency Domain • Separation in the Time Domain • Separation in Space
25.7 Summary and Conclusions 25-19
References 25-20

Kirsten Matheus
NXP Semiconductors

25.1 Introduction

The convenience of true mobility offered by wireless connectivity is the main driver behind its widespread acceptance. While cellular systems like GSM or UMTS require huge upfront investments and extensive infrastructure, the commercial and industrial deployment is more appealing when systems function on a smaller scale and do not require costly frequency licensing or infrastructure; systems like Bluetooth, IEEE 802.11 wireless local area networks (WLANs), or IEEE 802.15.4/ZigBee.

These systems, each having successfully been developed for a specific purpose, are now being investigated for use cases in industrial environments, where wireless connectivity has manifold advantages.

First of all, less cabling is needed. This normally results in a significant reduction of material costs and installation time. Wireless technologies might even enable specific setups otherwise impossible to realize. Maintenance costs can also be reduced, when replacements no longer need to be done where chemicals, vibrations, or moving parts would gradually damage cables . Moreover changes in the layout of the factory floor and reconfiguration are easier to realize [WMW05]. The actual

tasks performed on the factory floor with help of wireless transmissions are plentiful. They range from monitoring, diagnosis, tracking of parts or vehicles, over commissioning (task assignment) to centralized or even autonomous motion control [INS07].

Depending on the task, different requirements must be met. They have to be investigated carefully before choosing the most suitable technology (possibly not only one). TCP/IP communication, e.g., has no or soft real-time requirements while a Fieldbus system will have real-time requirements in the range of 10–100 ms [INS07].

The main criteria are generally throughput, delay, and reliability. In addition, cost, power consumption, security, and, last but not least, availability can be important issues. From the technologies under discussion, Bluetooth is a typical wireless personal area network (WPAN) representative. Bluetooth, which is inexpensive, consumes relatively little power, is small in size, and supports voice and data services. The different IEEE 802.11 variants are WLAN representatives that provide comparably high user-data rates at the cost of a higher battery power consumption. The limited range makes IEEE 802.15.4/ZigBee also rather a WPAN representative. It is limited to quite small data rates, but is at the same time optimized for long battery life.

Note that in industrial environments the radio conditions can be specific. This is particularly owned to the fact that on factory floors many metal objects and/or metal walls can be found. On one hand metal shields radio transmissions, while at the same time causing, respectively, more reflections. Measurements performed in industrial environments thus show smaller path loss coefficients* and noteworthy multipath propagations, which the systems need to be able to handle. To be able to handle the latter is a matter of implementation. There is no indication though that that cannot been achieved. A further speciality on the factory floor is electro magnetic interference from the machinery. It is found to drop off rapidly in the (considered) frequency range above 1 GHz and thus not considered further, though it can have significant impact in lower frequency bands. Note that a certain probability nevertheless always remains that a wireless transmission fails completely (e.g., because of complete shielding). Systems requiring a certain amount of data rate within a strict time window, e.g., because they are safety related, should not be wireless.

This chapter investigates the performances of WLAN and WPAN technologies from a radio network perspective. The radio network perspective includes that not only the hazards of the physical transmission (echoes, etc.) impact the performance but also the existence of other networks transmitting in the same space and frequency range. That means that in addition to the system inherent parameters also factors like unit density, traffic demand, mobility, environmental changes during deployment, interference, frequency range, etc. play a role. Thus both, the individual link performance and the overall network capacity, should be optimized. Higher protocol layers (e.g., TCP/IP, Fieldbus systems) are not part of the investigations. For an introduction into that topic please refer to [WMW05].

To outline the investigation and this chapter first describes in Section 25.2 basic differences between WLAN, WPAN, cellular ad hoc networks, and wireless sensor networks (WSNs). In Sections 25.3 through 25.5 the technologies Bluetooth, IEEE 802.11, and IEEE 802.15.4/ZigBee are described in more detail. Each of these sections provides technical background on the technology as well as investigations on the performance of the systems and their suitability for industrial applications/factory floor environments. Section 25.6 shows how the systems—all can be used in the same frequency band—coexist. Section 25.7 provides a summary and the conclusion.

* Published measurement results are 1.1 in a pulp mill [Rug04], 1.7 in an office building [BW00] instead of the expected 2.

25.2 WLAN, WPAN, WSN, and Cellular and Ad Hoc Networks

The expressions WLAN, WPAN, and cellular and ad hoc networks are commonly used, often though without consistency and/or precision. In industrial contexts additionally WSN play a role and are also not always clearly distinguished from WLANs and WPANs. In the following, a clarification of the terminology is given.

WLAN: A wireless LAN has the same functionality as a wired LAN with the difference that the wires are replaced by air links. This means that within a specific area (home, office, hot spot), intercommunication between all devices connected to the network is possible. WLANs are associated with data communication and "large" data rates. The definition of WLAN says nothing on how the network is organized. Often an infrastructure of mounted access points (APs) enables wireless access to the wired LAN behind the APs thus representing a cellular network structure. Nevertheless, a wireless LAN can also function on an ad hoc basis.

WPAN: In a WPAN also all devices are interconnectable. The difference is that all units are somehow associated to someone or something (a person, a robot, a specific shared or public devices). (W)PANs are associated with smaller scales concerning distance as well as amount of traffic. A PAN can consist of a variety of devices and can even comprise different technologies. The type of traffic is not decisive. In a PAN, data as well as voice communication can prevail.

While you move *within* the WLAN, you can generally move *with* your WPAN. This means that several independent WPANs possibly coexist in the same area, each being self-sufficient without any infrastructure. Because of the lack of central control, the overall network perspective is ad hoc based, though each individual PAN can be centrally organized.

WSN: In contrast to LANs or PANs, WSNs are not focussed on individual traffic between individual units but all devices are unified in the same task [Wil08]. Traffic is mainly uplink from sensor to central unit and it is often transmitted without requesting acknowledgments [ACKs]. Of the investigated technologies, IEEE 802.15.4/ZigBee is most suitable for WSNs. Still, while some results presented here might be transferable to WSNs, the focus is on WLANs and WPANs.

More decisive for the radio network performance of a technology (than the range or size of an individual network) is whether how a network is organized; with far-reaching consequences to system design, network optimization, etc. [MZTM03]. The difference between cellular and ad hoc networks is visualized in Figure 25.1. As can be seen, there are several steps that lead to ad hoc networking: A "pure" ad hoc network does neither employ any infrastructure nor a specific unit (like AP or base station) for the organization of coverage, synchronization, and services. Nevertheless a network can be "ad hoc," independent from whether it supports single or multihop communication.

It can be seen that WLAN technologies like IEEE 802.11 in the infrastructure mode (using the point coordination function [PCF]) are in the same classification as typical "cellular" systems like GSM or UMTS/WCDMA; all are based on infrastructure and use a specific unit for central control. Concerning the architecture, it would thus be correct to call these WLAN systems cellular systems. Despite this, there are distinct differences with regard to coverage. The "wide" (instead of "local") area coverage of cellular systems like GSM, has caused cellular systems to be associated with complete coverage and access everywhere, even though this is not correct. The description wireless wide area network (WWAN) gives a better idea of the difference to WLAN systems.

Cellular systems are also associated with voice traffic, though this is also not decisive. More important is that WWAN technologies are designed to support user mobility (and roaming) up to very high velocities, while WLAN systems rather support stationary or portable access. Industrial setups are generally limited to a specific area. Deploying a WWAN technology would represent an overkill and

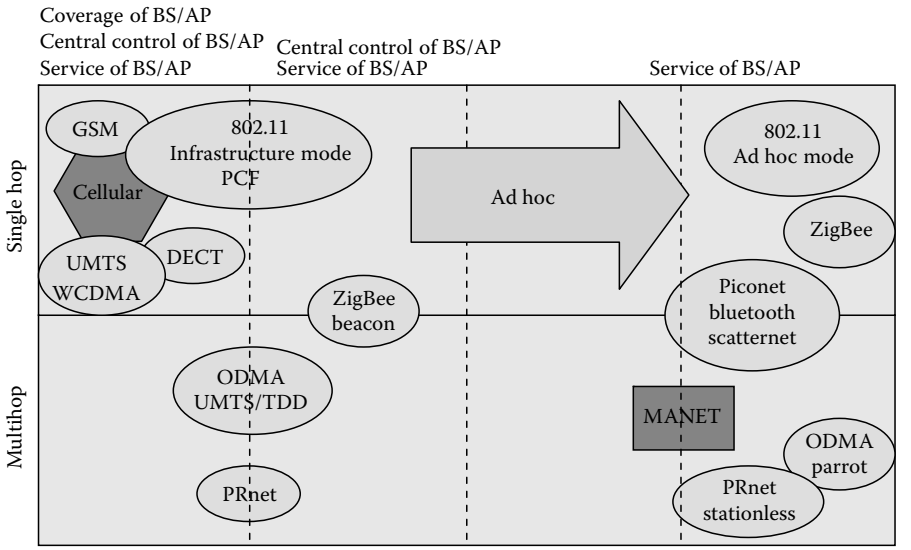


FIGURE 25.1 Classification of wireless technologies between cellular and ad hoc; MANET stands for Mobile (or Multihop [JIC99]) Ad hoc NETwork (designed for Internet applications [IET]), PRnet stands for Public Radio network [Mal01], ODMA for opportunity driven multiple access [3rd99,RBM02]; a Bluetooth scatternet means that several Bluetooth piconets are interlinked; BS = base station, AP = access point.

is—because of the licensing regulations and costs for extensive infrastructure—often not economic. WWANs are thus excluded from further investigations.

For the discussion of the radio network performance, most important differences between ad hoc and cellular systems are discussed in the following:

- As cellular networks are systematically laid out, the minimum distance to the next co-channel interferer (i.e., the next uncoordinated user transmitting at the same time on the same frequency) is generally controllable, known, and fixed. In contrast, in ad hoc networks the next co-channel interferer can be very close from one moment to the next, without any possibility to influence the situation (see Figure 25.2).
- Centralized control in cellular networks allows for effective and fair distribution of resources, because of the accurate knowledge of the overall traffic demand and the possibility of a more global resource management. For ad hoc networks or several coexisting WPANs, knowledge of overall traffic demand is generally not available and the systems compete for the resources.

25.3 Bluetooth Technology

25.3.1 Technical Background

Bluetooth technology (BT) is first of all a cable replacement technology aiming at effortless wireless connectivity in an ad hoc fashion. It supports voice as well as data transmission [Haa98,BS00,Bis01, Blu03,Blu05]. Its desired key associations are easy to use, low in power consumption, and low in cost, with the aim that the Bluetooth functionality is integrated in as many devices as possible.

To meet these goals, the BT special interest group placed the technology in the unlicensed ISM-band at 2.4 GHz. This allows close to worldwide deployment without needing to pay fees

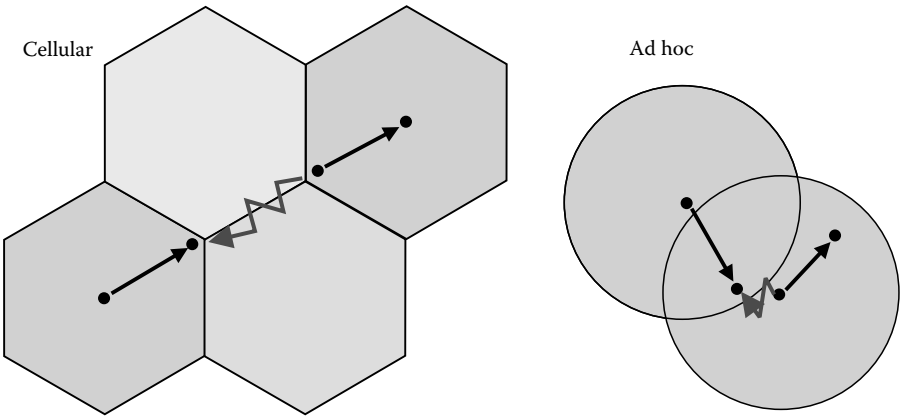


FIGURE 25.2 Closest next co-channel interferer in cellular and ad hoc networks. (From Matheus, K. *Industrial Information Technology Handbook*. Ed. R. Zurawski, CRC Press, Boca Raton, FL, 2004. With permission.)

for frequency licensing. Nevertheless, it requires complying to the respective sharing rules.* As a consequence, Bluetooth performs a rather fast frequency hopping (FH) over 79 carriers of 1 MHz bandwidth each, such that every Bluetooth packet is transmitted on a newly chosen frequency (which results in a nominal hop rate of 1600 hops/s). To further reduce cost and support the distribution of the Bluetooth technology, the Bluetooth specification is an open standard that can be used without needing to pay for the use of its key patents, on the term that the strict qualification procedure is passed.†

The latter is to ensure the acceptance of the technology. For the same purpose, the specification contains application profiles. The profiles ensure compatibility up to the application layer, thus enabling units of different manufactures to interoperate. The most important characteristics of the physical layer are as follows: The data is GFSK modulated at 1 Mbps and organized in packets consisting of access code, header, and payload. The employment of forward error correction (FEC) for the payload is optional. Interpacket interleaving is not performed. This allows for lower chip prices, because memory can be saved.

Bluetooth uses a master–slave concept in which the unit that initiates a connection is temporarily assigned "master" status (for as long as the connection is up). The master organizes the traffic of up to seven other active units, called "slaves" of this "piconet." From the master's device address the identity of each piconet and with it the FH sequence are derived. The header of a packet contains the actual addressee, the length of the packet, and other control information. Note that within one piconet the slave can only communicate with the master (and not directly with the other slaves) and this—in case of data connections—only after having been asked (i.e., "polled").‡

The channel is organized in a TDMA/TDD [GL00] scheme (see Figure 25.3). It is partitioned into 625 μs time slots. Within this slot grid the master can only start transmission in the odd-numbered slots while the slaves can only respond in even-numbered ones. When a unit is not already in one of the specific power save modes (sniff, hold, park), the slotting is power-consumption friendly, because

* For the United States [Fed02,Part 15] for Europe [Eur00a], for Japan [Min02].

† The Bluetooth specification has also been adopted by the IEEE as IEEE 802.15.1.

‡ Every BT unit can simultaneously be a member of up to four piconets (though it can be master in only one of them). A formation in which several piconets are interlinked in that manner is called "scatternet." Aspects like routing, which play an important role in scatternets, are not covered in this chapter. The chapter focuses on the properties of a single or multiple independent piconets.

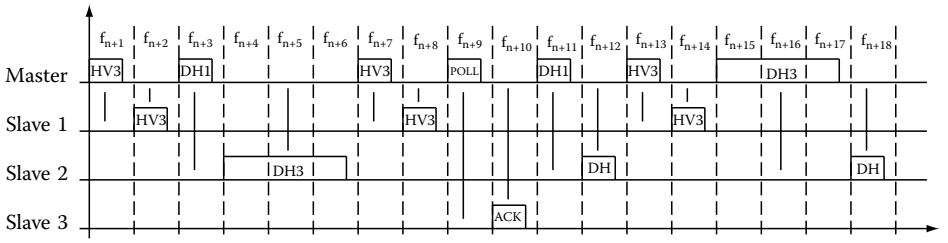


FIGURE 25.3 Example slot occupation within one piconet consisting of master and three slaves; to slave 1 there is an SCO-link, to slave 2 (best-effort) traffic is transmitted in both directions, slave 3 has currently nothing to send (but has to respond to the POLL-packet with an ACK); during the transmission of multislot packets the frequency is not changed. (From Matheus, K. *Industrial Information Technology Handbook*. Ed. R. Zurawski, CRC Press, Boca Raton, FL, 2004. With permission.)

every unit has to listen only during the first 10 μs of its receive slot whether there is a packet arriving (and if not, can "close down" until the next receive slot*). This means it needs to listen into the channel only $10\,\mu s/(2 \times 625\,\mu s) = 0.8\%$ of the time, during an active connection in which no packets are sent. Yet another facet to the long battery life is the low basic transmit power of 0 dBm (resulting in a nominal range of about 10 m). Bluetooth can also be used with up to 20 dBm transmit power. This results in a larger range but requires the implementation of power control to fulfill the FCC sharing rules.

Bluetooth provides two in principle different types of connections: ACL (asynchronous connection-less) links foreseen for data transmission and SCO (synchronous connection oriented) links foreseen for speech transmission.

For ACL-links there are six packet types defined. The packets occupy either one, three, or five (625 μs-) time slots and their payloads are either uncoded (called DH1, DH3, DH5) or protected with a 2/3-rate FEC using a (15,10) shortened Hamming block code without any interleaving (called DM1, DM3, DM5). An automatic repeat request (ARQ) scheme initiates the retransmission of a packet in case the evaluation of the cyclic redundancy check included in each ACL-payload shows inconsistencies. This secures error-free reception of the transmitted information. Table 25.1 gives an overview on the throughput values achievable with ACL-connections. The maximum (unidirectional) Bluetooth throughput is 723 kbps.

As speech transmission is delay sensitive, the original SCO-links support three different packet types that are transmitted at fixed intervals. These types were designed to transport CVSD (continuous variable slope delta) encoded speech at 64 kbps. The packet types occupy always just one

TABLE 25.1 Throughput Values for ACL-Connections

				Unidirectional Throughput		Bidirectional Throughput	
Name	No. of Slots	FEC?	Max. No. of User Bytes	Forward	Reverse	Forward	Reverse
DH1	1	no	27	172.8k	172.8k	172.8k	172.8k
DH3	3	no	183	585.6k	86.4k	390.4k	390.4k
DH5	5	no	339	723.2k	57.6k	433.9k	433.9k
DM1	1	2/3	17	108.8k	108.8k	108.8k	108.8k
DM3	3	2/3	121	387.2k	54.4k	258.1k	258.1k
DM5	5	2/3	224	477.8k	36.3k	286.7k	286.7k

Note: The reverse link in the unidirectional case transmits DH1 or DM1 packets, depending on whether the forward link uses a DH or DM packet type.

* This is quite different from channel access schemes like CSMA as used in IEEE 802.11 (see Section 25.4). Unless asleep, IEEE 802.11 always has to listen into the channel.

(625 μs-) time slot only, but they are differentiated by their payload FEC. The packet payloads are either unprotected (called HV3) or 2/3 rate FEC-encoded (HV2) or protected with a 1/3-rate repetition code (HV1). For an HV3 connection, a packet is transmitted every sixth slot (see Figure 25.3), for HV2 every fourth slot, and for HV1 every second slot (meaning that with one HV1 connection no other traffic can be transmitted in the piconet). Up to the Bluetooth Specification 1.1 [Blu99] there was no ARQ-scheme for SCO-links. In case of an erroneous reception of the packet overhead, the SCO-packet was replaced by an erasure pattern. In case noncorrectable bit errors occurred in the payload only, these errors were forwarded to the speech decoder. The Bluetooth Specification 1.2 [Blu03] includes an "enhanced" SCO-link. This link allows a very flexible deployment of the SCO-link, providing for a reserved bandwidth for several transmission rates and a limited number of retransmissions.

To further improve coexistence with other systems in the ISM-band, Bluetooth Version 1.2 includes the possibility to perform adaptive FH (AFH), i.e., to exclude interfered carriers from the hop sequence. With AFH the nominal hop rate will be halved, because the specification has been changed such that the slave responds on the same frequency as on which it received the packet from the master [Blu03]. The Bluetooth specification supports security by authentication and encryption.

Available publications on Bluetooth's radio network performance and its suitability for industrial environments generally use the Bluetooth 1.x versions as a basis. Nevertheless, later Bluetooth specifications exist. Bluetooth 2.0 + EDR (Enhanced Data Rate) allows for double or triple transmission rates by modulating more bits into one symbol, i.e., by using a π/4DQPSK (1446.2 kbps) or an 8DPSK (2169.6 kbps) instead of the GFSK. The standard is backward compatible and decides on the modulation scheme depending on the availability and link quality. Bluetooth 2.1 + EDR (published in August 2007) enables, among others, "Secure Simple Pairing" (which deploys, if available, near field communication technology) and an extended sleep mode for improved power saving. Latest standardization activities foresee a high speed channel, whose physical layer is based on ultra wide band and whose MAC is based on the WiMedia MAC. A transmission in the 6–9 GHz domain is discussed as well as transmission rates up to 480 Mbps.

25.3.2 Performance

On the factory floor, Bluetooth can be used as a wireless add-on to wired systems or as a replacement of existing cabling. It can cover machine-to-machine communication, wireless/remote monitoring, or tracking and some type of positioning of moving entities [GGH01,Bea02]. [BW00] and [Rug04] investigate the performance of a single Bluetooth link in a factory environment. The bit error rate (BER) turned out to be somewhat worse there than in an office environment. Nevertheless, the technology was seen as suitable for the use case.

Considering the comparably short range of Bluetooth and the likely association to a specific unit (represented by a machine/person/task), it is furthermore possible that several independently active Bluetooth piconets coexist and overlap in space. The use of FH helps to mitigate the effects of interference among these piconets. When assuming more or less time synchronized piconets, a worst-case approximation of the loss rate can be made with Equation 25.1. It calculates the probability $P(x, n)$ that of x other piconets n hop onto the same frequency as the considered piconet:

$$P(x,n) = \binom{x}{n}\left(\frac{1}{79}\right)^{n}\left(\frac{78}{79}\right)^{x-n}. \tag{25.1}$$

The probability that at least one of the other x piconets transmits on the same frequency is then $P(x) = 1 - P(x, 0)$. The smaller the number of interfering piconets, the better the approximation

offered by this approach, as for larger numbers, the distances to some of the interferers are likely to be too large to be harmful.

In [ZSMH00,Zür00,MZTM03], thus a more sophisticated approach has been chosen and Bluetooth–Bluetooth coexistence results have been obtained with help of detailed radio network simulations that include traffic, distribution, and fading models as well as adjacent channel effects. All results have been obtained for an office of $10 \times 20\,m^2$, assuming an average master–slave distance of 2 m. Naturally a factory floor is likely to be significantly larger than $10 \times 20\,m^2$ and to contain more metal objects. Still, the range Bluetooth is comparably small, so that it is less impacted by multipath propagation than WLAN technologies (see Section 25.4.2). On the factory floor it is thus possible to place the Bluetooth units with almost the same density as in the investigated office scenario without loss in performance. As a factory floor is larger than the investigated office, the overall number of piconets that can be used simultaneously on the factory floor is larger too. Additionally, location and traffic of the factory floor units are likely to be more predictable. Directive antennas also help to improve the performance. The results of the aforementioned publications thus give a good idea of what performance is achievable:

- A $10 \times 20\,m^2$ room supports 30 HV3 simultaneous speech connections with an average packet loss rate of 1% (a limit that still allows for acceptable quality).
- HV3 packet types are preferable to HV2 and HV1. The subjective quality will not increase with additional payload coding. Using a coded HV packet just increases (unnecessarily) the interference in the network and the power consumption.
- 100 (!) simultaneous WWW-sessions (bursty traffic with an *average* data rate of 33.2 kbps each) in the $10 \times 20\,m^2$ size room result in a degradation of the aggregate throughput of only 5%.
- Maximum aggregate throughput in the room is 18 Mbps (at 50 fully loaded piconets). These piconets then transmit at a unidirectional data rate of 360 kbps.
- Long and uncoded packets are preferable to shorter and/or coded ones. It takes 60 interfering piconets using the same packet type, 10 interfering HV1 connections (worst case), or a link distance of 27 m (which is far beyond the envisioned range of 10 m) before another packet type yields a larger throughput than DH5 [MZTM03]. It is advisable not to use the optional FEC (DM-packet types). As the coding is not appropriate to handle the almost binary character of the Bluetooth (ad hoc) transmission channel,* the additional power (and bandwidth) that would be needed for the coding can be saved.

Bluetooth is inexpensive and consumes significantly less power than IEEE 802.11 systems. The ACL-link is reliable with best-effort traffic (with a maximum throughput of 723 kbps). The SCO-link has reserved bandwidth though the packets might contain residual bit errors (even when using the enhanced SCO-link). In principle, Bluetooth is very robust against other Bluetooth interference and good performance can be achieved even in very dense environments. Note that customized implementations, which cannot be based on existing profiles, might be difficult to realize, as the regulations do not allow the implementation of proprietary solutions. The specification of new profiles though can be quite time consuming.

* The reasons are manifold. Without interference, the channel varies already due to hopping over 79 relatively narrowband channels. Additionally, with the wavelength used in Bluetooth even small changes in position can cause large changes in the received signal strength. When there is interference the effect becomes more pronounced. The existence or nonexistence of a close co-channel interferer can make the channel change from very good to very bad within the fraction of a moment (see also Figure 25.2).

25.4 IEEE 802.11

25.4.1 Technical Background

IEEE 802.11 comprises a number of specifications that define the lower layers (mainly the physical [PHY] and medium access control [MAC] layers) for WLANs [OP99,Ins99a,Ins99b,Ins03,Ins05a]. Being part of the IEEE 802 group means that an interface can be used (IEEE 802.2) to connect to the higher layers, which are then not aware of the—with IEEE 802.11 wireless—network that is actually transporting the data. The key intentions of IEEE 802.11 are thus to provide a high throughput and continuous network connection like available in wired LANs.

To encourage the wide employment of the technology, the use of IEEE 802.11 does not incur frequency licensing fees. IEEE 802.11 foresees either infrared (IR), transmits in the unlicensed ISM-band at 2.4 GHz (like Bluetooth) or in 5 GHz bands that are license exempt in Europe and unlicensed in the United States (UNII bands). In contrast to Bluetooth, the companies holding key patents within IEEE 802.11 can charge developers of IEEE 802.11 products for using the patents "on reasonable terms" [Ins02,Clause 6b].

In principle it is possible to have an IEEE 802.11 WLAN consisting of mobile stations (MS) only. It is more likely though that IEEE 802.11 is used as a wireless access technology to a wired LAN to which the connection is made by IEEE 802.11 APs. Should the AP only employ the distributed coordination function,* the MAC layer supports collision avoidance by employing carrier sense multiple access (CSMA). This means that before transmitting a packet the respective unit has to listen for the availability of the channel.† If the channel is sensed free after having been busy, the unit waits a certain period (called DIFS) and then enters a random back-off period‡ of

$$\underbrace{\text{random}\left(0 \ldots \min\left(2^{n_{\text{PHY}}+n_{\text{r}}}-1, 1023\right)\right)}_{\text{CW}} \times t_{\text{slot}} \tag{25.2}$$

where

n_{PHY} is a parameter depending on the type of physical layer chosen
n_{r} is the index of the retransmission of the packet
t_{slot} represents the slot duration

CW stands for contention window (with $\text{CW}_{\min} = 2^{n_{\text{PHY}}} - 1$). If the channel is available after this time period, the unit transmits its packet (consisting of PHY header, MAC header, and payload). Upon a correct reception, the addressee responds with an ACK-packet a short period (called SIFS) later (see also Figure 25.4). The realized ARQ mechanism ensures reliable data.

Obviously, the IEEE 802.11 WLAN MAC concept was designed for best-effort data traffic. This was fair enough for the use cases the system had originally been designed for. Services for which strict delay requirements exist, like speech, were not supported well by the basic IEEE 802.11 specifications. QoS is now nevertheless addressed in the IEEE 802.11e specification [Ins05b] (see also Table 25.3). High-priority traffic can access the channel with in average shorter back-offs than low-priority traffic. This basic QoS function is even mandatory in the commercial "WiFi" use of IEEE

* This is likely and assumed in the following of this chapter. In theory, the standard also provides additionally the use of a centralized PCF.

† The implementor can choose, whether the units react (a) on just other IEEE 802.11 traffic, (b) on just other IEEE 802.11 traffic above a certain receive signal strength, or (c) on any signal above a certain receive signal strength [Ins99b, Section 18.4.8.4], (d) has been found to give the best performance results in interfered environments [KAT06].

‡ The random back-off period is entered only when the channel was busy before. Else, the unit will transmit at once after DIFS.

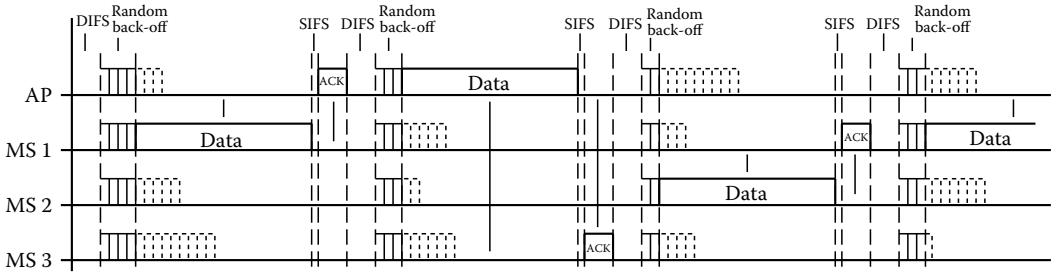


FIGURE 25.4 Principle time behavior of IEEE 802.11 under the distributed coordination function; note that the random back-off timer from units *not* transmitted next continues after the next DIFS with the remaining number of slots. (From Matheus, K. *Industrial Information Technology Handbook*. Ed. R. Zurawski, CRC Press, Boca Raton, FL, 2004. With permission.)

802.11. Nevertheless, this specification allows QoS in a single network only. Restriction owing to interference of coexisting networks cannot be compensated for by this QoS approach.

The IEEE 802.11 MAC concept includes an optional mechanism to solve the hidden terminal problem. Whether this "ready-to-send/clear-to-send" (RTS/CTS) packet exchange saves more bandwidth (due to avoided retransmissions) than it adds (due to additional overhead) depends on the terminal density and payload packet length [Bia00]. As the RTS/CTS mechanism does not have a large visibility, it will be assumed in the following that the RTS/CTS mechanism is not used.

IEEE 802.11 has a significantly larger power consumption than Bluetooth. Note that this is not only due to the higher transmit power (20 dBm in Europe, 30 dBm in the United States) but also due to the CSMA concept. IEEE 802.11 units not specifically in sleep status have to listen to the channel *all* the time (and not like Bluetooth just at the beginning of the receive slot). Simultaneously though the higher transmit power allows for a larger range of about 50 m indoors (with 20 dBm).

Originally, six different physical layer implementations has been specified. The IR mode, the FH spread spectrum mode, and the Packet Binary Convolution Code mode have not found real market acceptance though. They will therefore be excluded from further discussions.

DSSS: The Direct Sequence Spread Spectrum mode is (like Bluetooth) used in the 2.4 GHz ISM-band. The nominal bandwidth of the main lobe is 22 MHz. The transmit power reduction in the first and residual side lobes is supposed to be 30 and 50 dB, respectively (see also Figure 25.5 for a measured spectrum). In principle 11/13 (United States/Europe) center frequencies are available for the DSSS system. Nevertheless, using several systems in parallel requires a spacing of 25/30 MHz (United States/Europe), which consequently only allows three systems to be used simultaneously.

The DSSS mode comprises the original version (specified in IEEE 802.11) and a high rate extension (specified in IEEE 802.11b). In the original mode the chipping of the baseband signal is performed with 11 MHz, employing an 11-chip pseudorandom code (Barker sequence). For the 1 Mbps modulation rate a one bit DBPSK symbol is spread with the Barker sequence, for the 2 Mbps modulation rate a two bit DQPSK symbol is spread with the same sequence.* The introduction of the high rate extension IEEE 802.11b substantiated the market breakthrough of IEEE 802.11. For backward compatibility reasons, the PHY-header IEEE 802.11b uses the same 1 and 2 Mbps modulations as the plain DSSS mode. Note though that a shortened header of

* Note that in contrast to a typical CDMA system like UMTS, *all* users use the same spreading code.

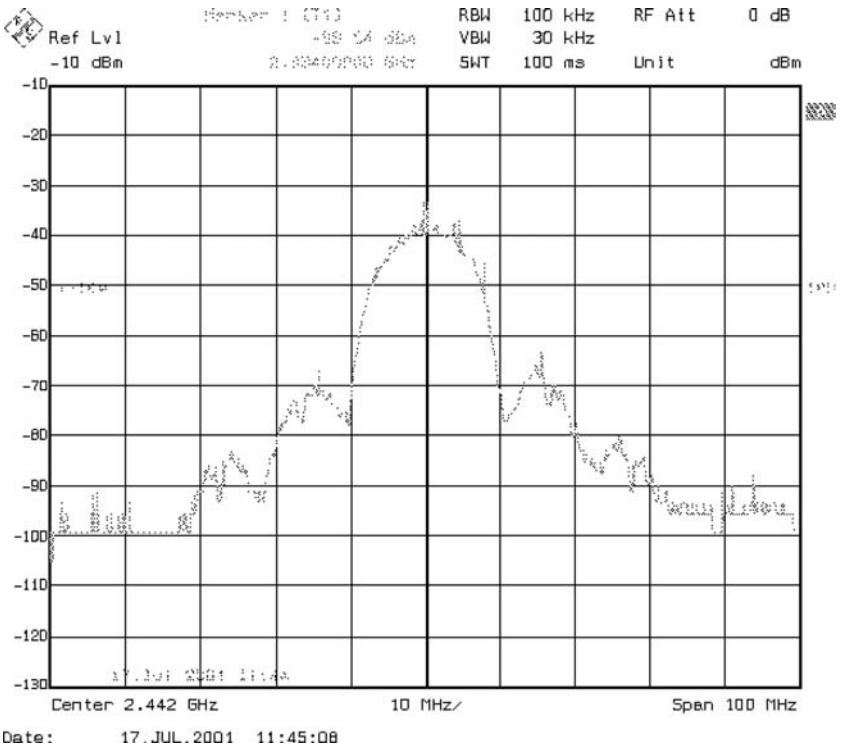


FIGURE 25.5 Measured spectrum of an IEEE 802.11b WLAN PCMCIA card. (From Matheus, K. *Industrial Information Technology Handbook*. Ed. R. Zurawski, CRC Press, Boca Raton, FL, 2004. With permission.)

96 μs can be used. For the IEEE 802.11b PHY-payload (consisting of the MAC header and the user data) a 5.5 and 11 Mbps complementary code keying (CCK) modulation is used. The CCK employs a variation of M-ary orthogonal signaling (complex Walsh/Hadamard functions) with eight complex chips in each spreading code word. For the 5.5 Mbps modulation rate 4 bits are mapped onto eight chips and for 11 Mbps 8 bits are mapped onto eight chips.

OFDM: The orthogonal frequency division multiplexing (OFDM) physical layer was originally designed for different 5 GHz bands (also referred to as IEEE 802.11a) but has now been adopted also for the 2.4 GHz band (as part of IEEE 802.11g). The parameters of the IEEE 802.11 OFDM PHY had at the time of standardization been harmonized with those of HIPERLAN/2.* Seven modi are defined ranging from BPSK with rate R=1/2 FEC (lowest modulation rate with 6 Mbps) to 64-QAM with rate R=3/4 FEC (highest modulation rate with 54 Mbps, see also Table 25.2). The OFDM technique is based on a 64-point IFFT/FFT, while only using 52 of the subcarriers (48 for user data, 4 for pilot carriers). The subcarrier spacing is $\Delta f = 20\,\text{MHz}/64 = 0.3125\,\text{MHz}$. Note that full OFDM symbols always have to be transmitted. This means that they possibly have to be filled up with dummy bits. To transmit one OFDM symbol $t_{\text{sym}} = 1/\Delta f + 1/4 \times 1/\Delta f = 4\,\mu\text{s}$ are needed, with the latter part representing the time

* Originally, HIPERLAN/2 was intended to be the WLAN technology for the European market while IEEE 802.11 was the pendant for North America. Owing to delays in the development, HIPERLAN/2 lost the chance to establish itself on the market, despite its better overall network performance (from which the user would have had the advantage of higher user-data rates). Publications on HIPERLAN/2 are, e.g., [KJSWW99,TM99,Eur00b,GL00,HZ00,LLM+00,LMP+00,MKJST00, Eur01a,Eur01b,KYW01,Mat04].

TABLE 25.2 Comparison of Different Achievable Maximum Throughput Rates in Mbps for the DSSS and OFDM IEEE 802.11 PHY Modes

Mode	Frq. Band	t_{slot}	SIFS	CW_{min}	t_{PHYh}	Modulation	ModRate [Mbps]	TP [Mbps] for PayBytes			
								60	576	1500	4061
DSSS	2.4GHz	20μs	10μs	31	192μs	DBPSK	1	0.30	0.80	0.91	0.97
						DQPSK	2	0.40	1.42	1.72	1.89
					96μs	CCK (QPSK)	5.5	0.67	3.14	4.26	4.97
						CCK (QPSK)	11	0.75	4.54	7.11	9.16
						BPSK, R1/2	6	1.51 (1.24/0.83)	4.57 (4.29 3.64)	5.37 (5.2/4.8)	5.76 (5.68/5.49)
						BPSK, R3/4	9	1.81 (1.44/0.91)	6.32 (5.81/4.67)	7.79 (7.43/6.64)	8.51 (8.35/7.96)
						QPSK, R1/2	12	2.02 (1.55/0.96)	7.86 (7.05/5.44)	10.0 (9.45/8.21)	11.2 (10.9/10.3)
OFDM	5 GHz (2.4 GHz)	9μs (9/20μs)	16μs (10μs)	15 (31)	20μs t_{sym} = 4μs	QPSK, R3/4	18	2.25 (1.70/1.01)	10.4 (8.97/6.53)	14.1 (13.0/10.8)	16.3 (15.8/14.4)
						16-QAM, R1/2	24	2.38 (1.76/1.03)	12.3 (10.3/7.22)	17.6 (15.9/12.7)	21.2 (20.2/18.1)
						16-QAM, R3/4	36	2.59 (1.86/1.07)	15.2 (12.3/8.14)	23.7 (20.8/15.6)	30.3 (28.4/24.3)
						64-QAM, R1/2	48	2.64 (1.89/1.07)	17.1 (13.7/8.7)	28.5 (24.3/17.5)	38.4 (35.4/29.3)
						64-QAM, R3/4	54	2.70 (1.92/1.08)	17.9 (14.2/8.90)	30.8 (26.0/18.3)	42.2 (38.6/31.4)

Source: Matheus, K. *Industrial Information Technology Handbook*. Ed. R. Zurawski, CRC Press, Boca Raton, FL, 2004. With permission.

used for the guard interval to combat multipath propagation. For synchronization, channel estimation and equalization a training sequence is transmitted, which consists of 10 repeated short and 2 repeated long OFDM symbols [Ins99c,vNAM+99].

DSSS-OFDM: This optional physical layer format of IEEE 802.11g combines the DSSS PHY with the OFDM PHY such that for the header DSSS is used while the payload employs OFDM (including the OFDM preamble).

Table 25.2 compares the theoretical maximum throughput values TP for the DSSS and OFDM IEEE 802.11 PHY versions for MAC data traffic. The maximum payload length is 4095 bytes (with has to include the 34-byte MAC header). 1500 bytes is the common length of an Ethernet packet (plus 34 bytes MAC header and check sum), 576 a typical length for a Web browsing packet and 60 bytes the length of a TCP ACK. The throughput TP is calculated as follows:

$$\mathrm{TP} = \frac{\mathrm{PayBytes} \times 8}{\underbrace{\mathrm{DIFS}}_{2t_{\mathrm{slot}} + \mathrm{SIFS}} + \underbrace{\frac{\mathrm{CW_{min}}}{2} \times t_{\mathrm{slot}}}_{\text{average back-off}} + t_{\text{data packet}} + \mathrm{SIFS} + t_{\mathrm{ACK}}}. \tag{25.3}$$

The durations needed to transmit the data packet $t_{\text{data packet}}$ and acknowledgment t_{ACK} vary depending on the physical layer chosen. For the DSSS mode they are calculated as follows:

$$t_{\text{data DSSS}} = t_{\mathrm{PHYh}} + \underbrace{\frac{34 \times 8}{\mathrm{ModRate}}}_{\text{MAC header}} + \frac{\mathrm{PayBytes} \times 8}{\mathrm{ModRate}}; \quad t_{\text{ACK DSSS}} = t_{\mathrm{PHYh}} + \frac{14 \times 8}{\mathrm{ModRate}}. \tag{25.4}$$

For the OFDM physical layer in the 5 GHz bands Equation 25.5 needs to be calculated. For the OFDM mode in the 2.4 GHz band, additional 6 μs signal extension have to be added to both. "Ceil" stands for the next larger integer.

$$\begin{aligned} t_{\text{data OFDM a}} &= t_{\mathrm{PHYh}} + t_{\mathrm{sym}} \mathrm{ceil}\left(\frac{16 + (34 + \mathrm{PayBytes}) \times 8 + 6}{\mathrm{ModRate}/12\,\mathrm{Mbps} \times 48}\right) \\ t_{\text{ACK OFDM a}} &= t_{\mathrm{PHYh}} + t_{\mathrm{sym}} \mathrm{ceil}\left(\frac{16 + 14 \times 8 + 6}{\mathrm{ModRate}/12\,\mathrm{Mbps} \times 48}\right). \end{aligned} \tag{25.5}$$

For small payload sizes (60 and 576 bytes), the maximum throughput values decrease drastically (2.7 and 17.9 Mbps maximum at 54 Mbps modulation rate). When considering Ethernet packets the highest theoretical throughput rates are 7.11 Mbps for IEEE 802.11b and 30.8/26.0 Mbps for the OFDM modes. Naturally, these wireless throughput rates are smaller than the wired ones (there 70–80 Mbps are possible), but at least for the higher modulation rates with 1500 bytes Ethernet packets the throughput values are reasonably good. Note that the real-life throughput values for IEEE 802.11b systems are still smaller than the theoretically possible ones: values around 5 Mbps have been measured [Mob01,MZ02]. This is because for the actual implementations used, higher protocol layers like TCP/IP cause additional overhead and delays.

For security, IEEE 802.11 WLANs support several authentication processes which are listed in the specification (none is mandatory).* Table 25.3 lists the IEEE 802.11 standardization activities involving IEEE 802.11.

* Neither Bluetooth nor IEEE 802.11 is renowned for their security concepts, and both have been criticized. Nevertheless, both systems have taken security aspects into consideration.

TABLE 25.3 Overview on Activities within IEEE 802.11

Group	Subject	Status
a m	PHY in the 5 GHz bands	Completed
b m	High rate mode in 2.4 GHz band	Completed
c	Extensions for specific MAC ooperations (bridging) IEEE 802.1D	Completed
d m	Supplements for new regulatory regions, raming extensions	Completed
e m	Enhancements for QoS	Completed
F	To achieve multi vendor AP interoperability	Withdrawn
g m	Enhancements of 802.11b data rates	Completed
h m	Extensions for channel selection for 802.11b	Completed
i m	Enhancements for security and authentication algorithms	Completed
j m	Enhancements for the use of 802.11a in Japan	Completed
k	Definition of radio resource management measurements	Conditional
l	Nonexistent	
m	Summarizes the extensions a, b, d, e, g, h, i, j in IEEE 802.11-2007	Completed
n	Higher throughputs using multiple input multiple output (MIMO)	Approved draft
o	Nonexistent	
p	For car-to-car or car-to-infrastructure communication	Ongoing
q	Nonexistent	
r	Fast roaming	Approved draft
s	Extensve Service Set mesh networking	Ongoing
t	Test methods and metrics recommendation	Ongoing
u	Interworking with non-802 networks (e.g. cellular)	Ongoing
v	Wireless network management	Ongoing
w	Protected management frames	Ongoing
x	Nonexistent	
y	3650–3700 MHz operation in the United States	Approved draft
z	Extensions to direct link setup	Started 2007

25.4.2 Performance

Next to aspects like individual link throughput, network capacity, and interference robustness, the transmission environment has to be taken into consideration when considering the use of IEEE 802.11 on the factory floor. As the scenarios envisioned for IEEE 802.11 were placed primarily in homes and offices, some differences occur when looking at the delay spread. While in homes and offices the delay spread is assumed to be <50 ns and <100 ns, respectively, it can be larger on factory floors. In [SMLW05], delay spreads with root mean square values between 10 and 200 ns were measured. [OP99] considers values of 200–300 ns. [Gor04]'s values go up to 500 ns.

In case of IEEE 802.11b a conventional rake receiver supports (only) about 60 ns delay spread in the 11 Mbps mode and 200 ns in the 5.5 Mbps version [vNAM+99]. When employing an IEEE 802.11b system with such a conventional receiver on a factory floor inter-symbol and/or inter-chip/codeword interference (ISI and/or ICI) are likely to degrade the performance. Nevertheless, with more complex receiver algorithms (like presented in [CLMK02,LMCW02]), IEEE 802.11b can compensate well for delay spreads of 1μs and even mobility of the user. Another option of course is to use IEEE 802.11a or g. Because of the guard interval inherent in the OFDM technology, delay spreads of several hundred nanoseconds can be easily supported without paying attention to the receiver algorithms implemented [vNAM+99].

The overall network performance (or interference performance, as discussed in Section 25.6)—in contrast to the individual link performance—is not that is often addressed as an issue. The publications that do exist do not at all support this attitude. In [Lin01] it is shown that co-channel interference with a carrier-to-interference ratio (CIR) of 5 dB still results in a packet loss rate of 10%–20% (BER = 10^{-5}) and that with a frequency offset of 5 MHz still CIR = 3 dB is required to achieve the same result. [Bia00,Sad00] present how the aggregate throughput in a single network decreases with the number of users, either due to hidden or exposed terminal problems or due to additional RTS/CTS overhead. With only 10 stations [Bia00] or a hidden node probability of 5% [Sad00], the system throughput is about halved (!) in case of 1500 byte payloads. Only when there are more than 25 stations is the RTS/CTS implementation justified, while the throughput is still reduced.

Thus, when installing IEEE 802.11 in a cellular fashion, some kind of frequency planning should be performed. Several algorithms have been published (see, e.g., [TMK01,HF04,CMTK07]); for IEEE

802.11a a mechanism has been standardized in IEEE 802.11h. The most relevant parameter for WLAN frequency planning is the number of mobile terminals that have to be served (assuming that this correlates with the demand for bandwidth). From it the optimum number of APs and their location (i.e., the distance between APs) can be determined. If it is desired that MS can seamlessly change between APs (due to mobile deployment) hand-over algorithms have to be added. [HF04] additionally recommends a centralized approach of load sharing between APs, for being able to double or triple the throughput.

IEEE 802.11 provides reliable best-effort traffic. For Ethernet traffic, the (theoretical) maximum transmission rates for IEEE 802.11a, g, and b are 30.8, 26.0, and 7.11 Mbps, respectively, and the technology is suitable for applications that require respective data rates. To achieve network capacity values three times these values (three parallel systems are possible for IEEE 802.11 b and g, for IEEE 802.11a networks capacity is less critical, owing to the larger frequency band at 5 GHz), appropriate receivers and sophisticated frequency planning should be used. IEEE 802.11 systems are not efficient for applications, in which small packet sizes prevail or when power consumption is a critical issue.

25.5 IEEE 802.15.4/ZigBee

25.5.1 Technical Background

The idea behind IEEE 802.15.4/ZigBee* was to create a very low cost, very low power, two-way wireless communication solution that meets the unique requirements of sensors and control devices needed in consumer electronics, home and building automation, industrial controls, PC peripherals, medical sensor applications, and toys and games. To allow long battery lives of 2 years and more (and thus minimizing the efforts in maintenance), the system supports low data rates at low duty cycles. Owing to its nominal range of 10 m, IEEE 802.15.4/ZigBee is generally considered a WPAN Technology [Bah02,Con03,Kin03]. Nevertheless, this is not as obvious as with Bluetooth. Owing to the large number of units an IEEE 802.15.4 network can support, one network can cover a very large area. Furthermore, IEEE 802.15.4/ZigBee is also often discussed in the context of WSN [KANS06,KAT06,BGSV07].

To encourage deployment also IEEE 802.15.4/ZigBee is placed in unlicensed frequency bands. Like Bluetooth and the 802.11b/g systems, IEEE 802.15.4/ZigBee can be used almost globally in the 2.4 GHz band. Additionally IEEE 802.15.4/ZigBee has been specified for the ISM-bands at 868 MHz in Europe and 915 MHz in North America. To comply with the respective sharing rules and to allow simple analogue circuitry, the system uses DSSS. Table 25.4 gives an overview on the respective physical layer parameters. Note that the maximum *user*-data rate is about 128 kbps, though smaller values are likely in case unsuitable parameters are used (see also Section 25.5.2).

The IEEE 802.15.4 standard allows for two different types of devices: The full function device (FFD) and the reduced function device (RFD). FFDs are unlimited in their communication, while RFDs can communicate with FFDs only. In an IEEE 802.15.4/ZigBee network one of the FFDs serves as a "PAN coordinator," who organizes the network, independent of its topology (which can be "star," "cluster tree," or "mesh"). Each network can handle up to 255 devices in case of a 16 bit address pad and even more in case of a 64 bit addressing. In contrast to Bluetooth ACL or IEEE 802.11 systems, ACKs are not mandatory for normal IEEE 802.15.4 data traffic, but are sent only upon request.

* Note, the technology comprises two specifications: The specifications published under IEEE 802.15.4 for the physical and MAC layer as well as the ZigBee specification, which covers the upper layers from network to application profiles [Cal03]. For the investigation topic of this chapter mainly the IEEE 802.15.4 part is relevant.

TABLE 25.4 IEEE 802.15.4 Parameters for the Different Frequency Bands

Frequency band	868 MHz	915 MHz	2.4 GHz
Location	Europe	North America	Worldwide
Number of center frequencies	1	10	16
Carrier spacing		2 MHz	5 MHz
Gross bit rate	20 kbps	40 kbps	250 kbps
Bit modulation	BPSK		16-ary orthogonal
Symbol rate	20 kbps	40 kbps	62.5 kbps
Spreading	15-chips M-sequence		32 chips PN code
Chip modulation	BPSK		O-QPSK
Chip rate	300 kchips/s	600 kchips/s	2 Mchips/s

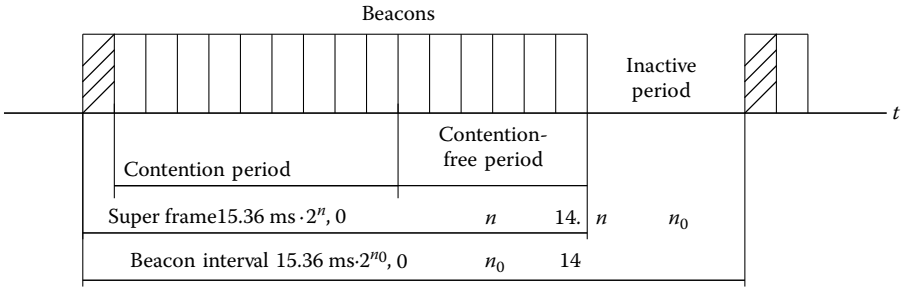


FIGURE 25.6 Optional IEEE 802.15.4/ZigBee beacon interval and super frame.

For medium access, there are in principle three possibilities: CSMA/CA in the unbeaconed mode, slotted CSMA/CA in the contention period of a beacon system, and guaranteed time slots (GTS) in the contention-free period of a beacon system. In the unbeaconed mode the star topology makes most sense. The PAN coordinator then needs to be continuously awake (and attached to a permanent power mode), while the other units have to be awake only when wanting to transmit something. Such a network design is most power effective for the other nodes, but allows for some applications only. In contrast, the mesh network with the beacon mode provides extended range and increases reliability as traffic can be rerouted around hindrances. This though reduces battery life and introduces latencies at the same time.

The beacon systems are defined by the beacon intervals and "super frames," which can be between 15.36 ms and 251.65 s (!) long (see also Figure 25.6). The length of the super frame determines the active period, which is divided into 16 equal time slots. If desired, the super frame is divided into a contention and a contention-free period (with a maximum length of seven slots), in which GTS can be allocated. The beacons are used for synchronization, for identification, and to describe the structure of the super frame. The used slotted CSMA/CA is somewhat different from the one in IEEE 802.11. First, the super frame time slots are further divided into back-off slots. The CSMA/CA is aligned to this slot grid (which helps to save power). Second, a packet which found the channel busy too often (the default value is five times) after the back-off timer expired, will be discarded.

For security IEEE 802.15.4/ZigBee provides authentication, encryption, and integrity service. The developer can choose between: no security, an access control list, and a 32 to 128 bit Advanced Encryption Standard encryption with authentication.

25.5.2 Performance

An IEEE 802.15.4 network can comprise a significant number of nodes, so the optimization of one IEEE 802.15.4 network is already a task not to underestimate.

[KAT06] found that for high offered loads in networks with mainly unacknowledged uplink traffic the stable throughput saturation is at 62%. Nevertheless, with increasing load the success probability decreases, so having the offered load between 35% and 60% is best.

Very important optimization parameters for IEEE 802.15.4 are the lengths of beacon interval and super frame. Low cycles mean low energy consumption but can result in high latencies and low bandwidths [LKR04]. Short frame sizes, in contrast reduce the available bandwidth by the number of beacons, increased collisions at the beginning of a frame and packets deferred to the next frame, when the remaining time is too short to fit it in [KAT06]. The highest throughput when using the shortest frame sizes and 50 byte packets is 38 kbps for one device [LKR04] or 32 kbps in case of more than one device. [PRML06] recommends to use low back-offs and short super frame orders only, if the resulting short channel access time is critical and the offered load is low. Otherwise the resulting large number of collisions is inefficient energy wise and leads to poor utilization of the channel. That means, for each network the beacon interval and super frame lengths need to be optimized individually, depending on the number of nodes and the desired throughput.

Several other aspects though have also been examined for network optimization. [SMLW05] find that spatial diversity with the help of mesh topology helps to increase the performance, as the latter is extremely location dependent. In [BGSV07] the polling is optimized for respective setups. In [HMBJ07] the main focus is on the hidden node problem. It is found that hidden nodes can significantly impact the performance within a network and that, depending on load, desired throughput and beacon interval, an optimum transmit power exists. [KANS06] introduce a queuing system, in which for high-priority traffic (such as alarm reports, PAN management messages, and GTS requests) a queue with different parameters (back-off, contention window length, etc.) is used.

None of the publications indicted that the use of IEEE 802.15.4/Zigbee for the intended use cases is critical; just care needs to be taken when designing the network.

25.6 Coexistence of WPAN and WLAN

Multiple wireless technologies will in the near future coexist: Within an enterprise, e.g., WLAN technologies could be used for flexible access to large corporate data bases while WPAN technologies handle specific tasks (and cellular systems the voice communication). Ideally, every (mobile) unit would connect effortlessly using whatever technology is most suitable at the time. The smooth handover between the technologies is thereby not the challenge one might expect. What causes problems is, when the deployed communication technologies are placed in the same frequency band and/or are linked by the application with each other (e.g., someone uses a Bluetooth headset with the mobile telephone).

As mentioned, all three investigated technologies can be used in the 2.4 GHz ISM band. All can be useful on the factory floor and might even be found in the same device or handle parts of the same application.

Numerous publications cover the mutual interference and performance impairments of Bluetooth and IEEE 802.11 (e.g., [Enn98,Zyr98,Fum01,GvDS01,How01,Mob01,MZ02]). Depending on the investigated scenarios, the assessment of the situation varies from "good reliability even in fairly dense environments" to "the effects of interference can be quite severe." A relative agreement exists in the parameters that determine the systems' performances: link distances (BT–BT, 802.11b–802.11b, BT transmitter–802.11b receiver, BT receiver–802.11b transmitter), traffic load, Bluetooth packet type, density of units, local propagation conditions.

It was found that IEEE 802.11b requires a CIR of about 10 dB to cope with a (narrowband) Bluetooth hop into its (wide) passband. IEEE 802.11b has the disadvantage that its back-off procedure was designed to optimize the IEEE 802.11b WLAN performance but not to handle external interferers: Each loss of a packet due to a collision with Bluetooth will increase the back-off window size by factor two (causing an unnecessary throughput reduction). Furthermore, the protocols overlaying WLAN often incorporate TCP, which includes the risk that packet losses on the air link are mistaken for network congestion, which then might initiate a slow start. In contrast, the main disadvantage of Bluetooth is that its transmit power is 20 dB below that of IEEE 802.11b. Another one is that the BT reverse link packet which contains the ACK is transmitted on a different frequency than the forward link packet. This increases the packet loss probability in case of (frequency static) IEEE 802.11b interference. The packet loss rate for Bluetooth $\mathrm{PLR_{BT}}$ then yields to

$$\mathrm{PLR_{BT}} = \mathrm{PLR_{forward}} + \underbrace{(1 - \mathrm{PLR_{forward}}) \times \mathrm{PLR_{header}}}_{\mathrm{PLR_{reverse}}}. \tag{25.6}$$

It would otherwise—were the forward and reverse link to use same hop frequency—be $\mathrm{PLR_{BT}} \approx \mathrm{PLR_{forward}}$ (depending on the IEEE 802.11b system load).

In [PRML06], the coexistence between IEEE 802.11 system and IEEE 802.15.4 has been investigated. Here it was found that, owing to the (same) difference in transmit power, IEEE 802.15.4 can be heavily affected by 802.11, but that vice versa effect can hardly be noticed. In contrast to Bluetooth, IEEE 802.15.4 does not hop. If IEEE 802.15.4 is being used in the same frequency band as an interferer, the performance is impaired for as long as the interferer transmits. Bluetooth's FH represents a principle advantage here; also in the range performance. While the Bluetooth performance can be expected to degrade smoothly with increasing range [Haa03], IEEE 802.15.4 has been found to enter a "gray zone" (owing to the frequency selective channel), in which the performance is unpredictable, before the connections is lost [PRML06].

There are in principle three different approaches to assist otherwise interfering systems to coexist: separation in time, separation in frequency and separation in space.*

25.6.1 Separation in the Frequency Domain

1. IEEE 802.11b can be used with an improved transmit filter that reduces the interference power on the side lobe frequencies and thus enhances the separation on those carriers.
2. Bluetooth can perform AFH, i.e., exclude the most heavily interfered frequencies from its hop sequence. Note that in a lot of realistic situations, where the IEEE 802.11b and Bluetooth units are not in the same device, AFH is sufficient to combat the interference effects. AFH has thus become part of the Bluetooth Specification 1.2 [Blu03].
3. For IEEE 802.15.4 the channel allocation needs to be done carefully (ideally with an algorithm that automatically selects the least interfered channel). [PRML06] found that the center frequency of IEEE 802.15.4 should be shifted by at least 7 MHz compared to the center frequency of IEEE 802.11.

* The approaches "separation through code" (keyword "CDMA"), through the channel (keyword "MIMO"), or through the modulation ("I versus Q") allow to unlink several users of *the same* system. It is not obvious though to apply any of these latter methods to improve the coexistence of *different* systems, which is what is needed here.

25.6.2 Separation in the Time Domain

1. IEEE 802.11b carrier sensing algorithm could also consider Bluetooth and IEEE 802.15.4/ZigBee signals* as well as Bluetooth (and IEEE 802.15.4/ZigBee) could be extended with a carrier sensing algorithm [ZKJ00]. The principle problem of carrier sensing though is, that to be really effective, the transmitter has to sense the situation at the receiver correctly, i.e., the correlation between transmitter and receiver situation has to be high. In an uncoordinated WLAN–WPAN scenario, the hidden as well as the exposed terminal problems are likely to countermeasure any advantage there might be.
2. Joint scheduler can allot alternating transmit time shares to both systems in a (to be specified) fair way. This, of course, only works (and with AFH is necessary only) when both systems are in one device, or even on one chip. Then again, such a measure might be a necessity, when otherwise blocking would occur.

25.6.3 Separation in Space

1. Should IEEE 802.11b and Bluetooth or IEEE 802.15.4 coexist in the same unit, an intelligent antenna design and placement is necessary to optimize the isolation between the antennas and to prevent blocking. This will not reduce collisions but minimizes their impact by maximizing the CIRs for the two systems.†
2. Antenna diversity can help each of the technologies individually to improve its performance.

25.7 Summary and Conclusions

When deciding on a wireless technology to use, first the characteristics of the foreseen application have to be clarified: Are the units going to move with or within the network, is the mobility range small or large, at what speed do the units move, is access needed to a large data base or just locally, is battery life a critical issue, what maximum distance should the wireless link cover, what distance does it cover in average, etc.? As a next step, the existing technologies can be viewed for their applicability.

Bluetooth is a comparably power efficient WPAN technology. Like all wireless systems it cannot provide hard throughput guarantees. Bluetooth is nevertheless quite robust for best-effort traffic in coexistence environments. One hundred Bluetooth piconets can transmit at an average data rate of 95%·33.2 kbps in an area of $10 \times 20\,m^2$. Fifty fully loaded piconets can transmit at an average, unidirectional transmission rate of 360 kbps. Similar results are likely to be achieved on larger factory floors (provided the piconet density is comparable), as the disadvantage of a larger number of units can be outweighed by the more structured and predictable unit location. Bluetooth is the only of the discussed technologies that supports voice well in addition to data transmission.

---

* As has been mentioned before, the IEEE 802.11 specification already provides for the possibility that IEEE 802.11 senses and not transmits if other systems than IEEE 802.11 systems are active [Ins99b, Section 18.4.8.4]. This has been found to give the best results in interfered environments [KAT06]. Nevertheless, even this mechanism holds a principle problem: IEEE 802.11 will refrain from transmission when it senses that another, near by system *transmits*. To improve coexistence though IEEE 802.11 should refrain from transmission when nearby units *receive*.

† Helpful to minimize the interference power is of course also power control (when applied by the interfering unit). Nevertheless, even though it can be recommended to implement power control (let alone to save power [MZTM03]), in real-life situations it cannot be relied upon that the interfering unit can indeed live with less power.

IEEE 802.11 is a WLAN technology that enables higher data rates but is not too power-consumption friendly. Additionally, in a relatively dense network scenario the theoretical maximum aggregate throughput of three times 7.11/30.8 Mbps is very likely to be seriously impaired. To aid the WLAN performance on a factory floor, it is thus advisable to take the following two measures: Apply means to combat the increased delay spread (in case of IEEE 802.11b) and additionally (for all IEEE 802.11 systems) carefully plan the frequency layout and AP placement.

IEEE 802.15.4/ZigBee has been designed specifically for sensor data and control information at low data rates. IEEE 802.15.4/ZigBee supports long battery lives. The achievable throughput is significantly smaller than that of Bluetooth (or IEEE 802.11). It depends heavily not only on actual bandwidth demand but also on the chosen parameters for beacon interval and super frame setup. The maximum overall load was found to be stable at 62%, while being most efficient, if the load offered by the units was between 30% and 65%.

The best way to aid the coexistence of Bluetooth and IEEE 802.11b is to use Bluetooth with AFH. If this is not sufficient, hardware-related improvements and/or a common scheduler have to be added. As IEEE 802.15.4 and IEEE 802.11 are both non-hopping systems, their coexistence should be foreseen in a respective frequency planning for both systems.

References

[3rd99] 3rd Generation Partnership Project. Technical Specification Group Access Network. Opportunity Driven Multiple Access, December 1999. 3G TR 25.924 version 1.0.0.

[Bah02] V. Bahl. ZigBee and Bluetooth—Competitive or Complementary? ZigBee Alliance, September 2002.

[Bea02] D. Beaumont. More Bluetooth products but key profiles delayed. *Planet Wireless*, July 2002.

[BGSV07] M. Bertocco, G. Gamba, A. Sona, and S. Vitturi. Performance measurements of CSMA/CA-based wireless sensor networks for industrial applications. In *Proc. IEEE Instrumentation and Measurement Technology Conference* (*IMTC*), Warsaw, May 2007.

[Bia00] G. Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal on Selected Areas in Communications*, 18(3):535–547, 2000.

[Bis01] C. Bisdikian. An overview of the Bluetooth wireless technology. *IEEE Communications Magazine*, 39(12):86–94, December 2001.

[Blu99] Bluetooth Special Interest Group. Specification of the Bluetooth System, Version 1.1, December 1999.

[Blu03] Bluetooth Special Interest Group. Bluetooth 1.2 Core Specification, November 2003.

[Blu05] http://www.bluetooth.com, 2005.

[BS00] J. Bray and C. F. Sturman. *Bluetooth: Connect without Cables*. Prentice-Hall, Englewood Cliffs, NJ, 2000.

[BW00] U. Bilstrup and P.-A. Wiberg. Bluetooth in industrial environment. In *Proc. 2000 IEEE Workshop on Factory Communication Systems, WFCS'2000*, pp. 239–246, Porto, Portugal, 2000.

[Cal03] Ed. Callaway. Low power consumption features of the IEEE 802.15.4/ZigBee LR-WPAN Standard. Presentation slides, November 2003.

[CLMK02] M. V. Clark, K. K. Leung, B. McNair, and Z. Kostic. Outdoor IEEE 802.11 cellular networks: radio link performance. In *Proc. IEEE International Conference on Communications* (*ICC*), New York City, May 2002.

[CMTK07] T. M. Chan, K. F. Man, K. S. Tang, and S. Kwong. A jumping-genes paradigm for optimizing factory WLAN network. *IEEE Transactions on Industrial Informatics*, 3(1): 33–43, February 2007.

[Con03] Cambridge Consultants. Unleashing renenue with ZigBee. Presentation slides, 2003.

[Enn98] G. Ennis. Impact of Bluetooth on 802.11 Direct Sequence. Technical Report IEEE 802.11-98/319, IEEE, September 1998.

[Eur00a] European Telecommunications Standards Institute (ETSI). ETSI EN 300 328-1 V1.2.2, July 2000.

[Eur00b] European Telecommunications Standards Institute (ETSI). HIPERLAN Type 2; Data Link Control Specification; Part 1: Basic Data Transport Functions, November 2000.

[Eur01a] European Telecommunications Standards Institute (ETSI). HIPERLAN Type 2; Data Link Control Specification; Part 2: Radio Link Control (RLC) Sub-Layer, April 2001.

[Eur01b] European Telecommunications Standards Institute (ETSI). HIPERLAN Type 2; Physical (PHY) Layer, February 2001.

[Fed02] Federal Communications Comission (FCC). Code of Federal Regulations, 2002.

[Fum01] D. Fumolari. Link performance of an embedded Bluetooth personal area network. In *Proc. IEEE International Conference on Communications* (*ICC*), Helsinki, June 2001.

[GGH01] J. Green, R. Gear, and N. Harman. Bluetooth: Users, Applications and Technologies. Ovum Report, June 2001.

[GL00] A. C. V. Gummalla and J. O. Limb. Wireless medium access control protocols. *IEEE Communications Surveys and Tutorials*, 3(2), 2000.

[Gor04] P. Gorday. 802.15.4 Multipath. doc: IEEE 802.15-04-0337-00-004b, July 2004.

[GvDS01] N. Golmie, R. E. van Dyck, and A. Soltanian. Bluetooth and 802.11b Interference: Simulation Model and System Results. Technical Report IEEE802.15-01/195R0, IEEE, April 2001.

[Haa98] J. Haartsen. Bluetooth—The universal radio interface for ad hoc, wireless connectivity. *Ericsson Review*, 3:110–117, 1998.

[Haa03] J. Haartsen. ZigBee OR Bluetooth or ZigBee AND Bluetooth. Presentation slides, June 2003.

[HF04] A. Hills and B. Friday. Radio resource management in wireless LANs. *IEEE Communications Magazine*, 42(12): S9–S14, December 2004.

[HMBJ07] M. Harthikote-Matha, T. Banka, and A. P. Jayasumana. Performance degradation of IEEE 802.15.4 slotted CSMA/CA due to hidden nodes. In *Proc. 32nd IEEE Conference on Local Computer Networks*, Dublin, October 2007.

[How01] I. Howitt. IEEE 802.11 and Bluetooth coexistence analysis methodology. In *Proc. IEEE Vehicular Technology Conference* (*VTC*), Rhodes, May 2001.

[HZ00] J. Huschke and G. Zimmermann. Impact of decentralized adaptive frequency allocation on the system performance of HIPERLAN/2. In *Proc. IEEE Vehicular Technology Conference* (*VTC*), Tokyo, May 2000.

[IET] http://www.ietf.org/html.charters/manet-charter.html.

[Ins99a] Institute of Electrical and Electronic Engineering. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, September 1999. ANSI/IEEE Std 802.11.

[Ins99b] Institute of Electrical and Electronic Engineering. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band, September 1999. IEEE Std 802.11b-1999.

[Ins99c] Institute of Electrical and Electronic Engineering. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher-Speed Physical Layer in the 5 GHz Band, September 1999. IEEE Std 802.11a-1999.

[Ins02] IEEE-SA Standards Board Bylaws, September 2002. Approved by Standards Association Board of Governors.

[Ins03] Institute of Electrical and Electronic Engineering. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band, June 2003. ANSI/IEEE Std 802.11.

[Ins05a] http://grouper.ieee.org/groups/802/11/index.html, 2005.

[Ins05b] Institute of Electrical and Electronic Engineering. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements, November 2005.

[INS07] S. Ivanov, E. Nett, and S. Schemmer. Planning available WLAN in dynamic production environments. In *Proc. 7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems* (*FET'2007*), Toulouse, France, November 2007.

[JIC99] L. Ji, M. Ishibashi, and M. S. Corson. An approach to mobile ad hoc network protocol Kernel design. In *Proc. IEEE Wireless Communications and Networking Conference* (*WCNC*), New Orleans, LA, September 1999.

[KANS06] A. Koubaa, M. Alves, B. Nefzi, and Y.-Q. Song. Improving the IEEE 802.15.4 slotted CSMA/CA MAC for time-critical events in wireless sensor networks. In *Proc. Workshop of Real-Time Networks* (*RIN 2006*), *Satellite Workshop to* (*ECRTS 2006*), Dresden, Germany, July 2006.

[KAT06] A. Koubaa, M. Alves, and E. Tovar. A comprehensive simulation study of slotted CSMA/CA for IEEE 802.15.4 wireless sensor networks. In *Proc. 2006 IEEE International Workshop on Factory Communication Systems* (*WFCS*), Torino, July 2006.

[Kin03] P. Kinney. ZigBee technology: Wireless control that simply works. *Communication Design Conference*, San Jose, October 2003.

[KJSWW99] J. Khun-Jush, P. Schramm, U. Wachsmann, and F. Wenger. Structure and performance of the HIPERLAN/2 Physical layer. In *Proc. IEEE Vehicular Technology Conference* (*VTC*), pp. 2667–2671, Amsterdam, the Netherlands, Fall, 1999.

[KYW01] A. Kadelka, E. Yidirim, and B. Wegmann. Serving IP mobility with HIPERLAN/2. In *Proc. European Mobile Communications Conference* (*EPMCC*), Vienna, Austria, February 2001.

[Lin01] M. Lindgren. Physical Layer Simulations of the IEEE 802.11b Wireless LAN-Standard. Master's thesis, Luleå Technical University, Luleå, Sweden, 2001.

[LKR04] G. Lu, B. Krishnamachari, and C. S. Raghavendra. Performance evaluation of the IEEE 802.15.4 MAC for low-rate low-power wireless networks. In *Proc. 2004 IEEE International Conference on Performance, Computing, and Communications*, pp. 701–706, Phoenix, AZ, April 2004.

[LLM+00] H. Li, J. Lindskog, G. Malmgren, G. Miklos, F. Nilsson, and G. Rydnell. Automatic Repeat Request (ARQ) Mechanism in HIPERLAN/2. In *Proc. IEEE Vehicular Technology Conference* (*VTC*), Tokyo, Japan, May 2000.

[LMCW02] K. K. Leung, B. McNair, L. J. Cimini, and J. H. Winters. Outdoor IEEE 802.11 cellular networks: MAC protocol design and performance. In *Proc. IEEE International Conference on Communication* (*ICC*), New York, April–May 2002.

[LMP+00] H. Li, G. Malmgren, M. Pauli, J. Rapp, and G. Zimmermann. Performance of the radio link protocol of HIPERLAN/2. In *Proc. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications* (*PIMRC*), London, 2000.

[Mal01] D. A. Maltz. On-demand routing in multi-hop wireless mobile ad hoc networks. PhD thesis, School of Computer Science Carnegie Mellon University, Pittsburgh, PA, May 2001. http://reports-archive.adm.cs.cmu.edu/anon/2001/CMU-CS-01-130.pdf.

[Mat04] K. Matheus. Wireless local area networks and wireless personal area networks (WLANs and WPANs). In R. Zurawski, editor, *Industrial Information Technology Handbook*. CRC Press, Boca Raton, FL, 2004.

[Min02] Ministry of Telecommunications (MKK), Japan. RCR STD-33A, 2002.

[MKJST00] G. Malmgren, J. Khun-Jush, P. Schramm, and J. Torsner. 6:3 HIPERLAN Type 2—An emerging world wide WLAN standard. In *Proc. of International Symposium on Services and Local Access*, Stockholm, Sweden, June 2000.

[Mob01] Mobilian. Wi-Fi (802.11b) and Bluetooth: An Examination of Coexistence Approaches. http://www.mobilian.com, 2001.

[MZ02] K. Matheus and S. Zürbes. Co-existence of Bluetooth and IEEE 802.11b WLANs: Results from a radio network testbed. In *Proc. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications* (*PIMRC*), Lisbon, Portugal, September 2002.

[MZTM03] K. Matheus, S. Zürbes, R. Taori, and S. Magnusson. Fundamental properties of ad-hoc networks like Bluetooth: A radio network perspective. In *Proc. IEEE Vehicular Technology Conference* (*VTC*), Orlando, FL, September 2003.

[OP99] B. O'Hara and Al Petrick. *IEEE 802.11 Handbook: A Designer's Companion*. Standards Information Network IEEE Press, New York, 1999.

[PRML06] M. Petrova, J. Riihijärvi, P. Mähönen, and S. Labella. Performance study of IEEE 802.15.4 using measurements and simulations. In *Proc. IEEE Wireless Communications and Networking Conference* (*WCNC*), Las Vegas, NV, 2006.

[RBM02] T. S. Rouse, I. W. Band, and S. McLaughlin. Capacity and power analysis of opportunity driven multiple access (ODMA) networks in CDMA systems. In *Proc. IEEE International Conference on Communications* (*ICC*), pp. 3202–3206, New York City, 2002.

[Rug04] M. Ruggiero. Bluetooth in Industrial Environment, March 2004. http://www.expertmon.com/up_files/Bluetooth.pdf.

[Sad00] S. Sadalgi. A performance analysis of the basic access IEEE 802.11 wireless LAN MAC protocol (CSMA/CA), May 2000.

[SMLW05] D. Sexton, M. Mahony, M. Lapinsky, and J. Werb. Radio channel quality in industrial wireless sensor networks. In *Proc. Sensors for Industry Conference* (*SIcon/05*), pp. 88–94, Houston, TX, February 2005.

[TM99] J. Torsner and G. Malmgren. Radio network solutions for HIPERLAN/2. In *Proc. IEEE Vehicular Technology Conference* (*VTC*), pp. 1217–1221, Houston, TX, Spring 1999.

[TMK01] K.-S. Tang, K.-F. Man, and S. Kwong. Wireless communication network design in IC factory. *IEEE Transactions on Industrial Electronics*, 48(2):452–459, April 2001.

[vNAM+99] R. van Nee, G. Awater, M. Morikura, H. Takanashi, M. Webster, and K. W. Halford. New high-rate wireless LAN standards. *IEEE Communications Magazine*, 37(12):82–88, December 1999.

[Wil08] A. Willig. Recent and emerging topics in wireless industrial communications: A Selection. *IEEE Transactions on Industrial Informatics*, 4(2):102–124, May 2008.

[WMW05] A. Willig, K. Matheus, and A. Wolisz. Wireless technology in industrial networks. *Proc. IEEE; Special Issue on Industrial Communication Systems*, 93(6):1130–1151, 2005.

[ZKJ00] B. Zhen, Y. Kim, and K. Jang. The analysis of coexistence mechanisms of Bluetooth. In *Proc. IEEE Vehicular Technology Conference* (*VTC*), Tokyo, Japan, Spring 2000.

[ZSMH00] S. Zürbes, W. Stahl, K. Matheus, and J. Haartsen. Radio network performance of Bluetooth. In *Proc. IEEE International Conference on Communication* (*ICC*), New Orleans, LA, June 2000.

[Zür00] S. Zürbes. Considerations on link and system throughput of Bluetooth networks. In *Proc. IEEE Int. Symposium on Personal, Indoor and Mobile Radio Communications* (*PIMRC*), pp. 1315–1319, London, September 2000.

[Zyr98] J. Zyren. Extension of Bluetooth and 802.11 direct sequence interference model. Technical Report IEEE 802.11-98/378, IEEE, November 1998.

26

Hybrid Wired/Wireless Real-Time Industrial Networks

26.1 Introduction **26**-1
26.2 Relevant Industrial Networks **26**-3
Profibus DP and Profinet IO • DeviceNet and EtherNet/IP • IEEE 802.11 • IEEE 802.15.4
26.3 Implementation of Hybrid Networks **26**-6
Interconnections at the Physical Layer • Interconnections at the Data-Link Layer • Interconnections at Higher Protocol Layers • Wireless Extensions of Industrial Networks
26.4 IEEE 802.11-Based Extensions: Fieldbuses **26**-11
Extensions of Profibus DP • Extensions of DeviceNet
26.5 IEEE 802.11-Based Extensions: RTE Networks **26**-14
Extensions of Profinet IO • Extensions of EtherNet/IP
26.6 IEEE 802.15.4-Based Extensions **26**-16
Extensions of Fieldbuses • Extensions of RTE Networks
26.7 Conclusions **26**-17
References **26**-18

Gianluca Cena
National Research Council

Adriano Valenzano
National Research Council

Stefano Vitturi
National Research Council

26.1 Introduction

The past two decades have witnessed the ever-increasing adoption of advanced digital communication technologies in industrial automation and manufacturing environments. Field networks (also known with the term "fieldbuses") [1] are certainly the most popular solution in those scenarios. Since a long time, they are being used at the lowest levels of factory automation systems (shop-floor) that are often characterized by tight timing requirements. On the other hand, Ethernet networks, which were traditionally adopted as factory backbones to handle information flows between different areas of the plant, are now deemed suitable to support also the communication needs of distributed control applications down to the field level. This is mostly due to the noticeable enhancements concerning performance and determinism brought to this kind of networks over the years, which led, in the recent past, to the development of industrial Ethernet solutions purposely designed to support real-time communications [2]. It is worth noting that, in several cases, variants of such protocols have been defined that are able to support isochronous communications as well, like those required in demanding the motion control systems.

The above-mentioned scenario, however, is quickly evolving. Other kinds of communication technologies, i.e., wireless networks, are now very popular and related devices are widely available off-the-shelf at reasonable prices. Wireless communications are becoming more and more pervasive

everyday, and they are changing deeply many aspects of human life, by enabling new perspectives and opportunities that could hardly even be thought of only few years ago (e.g., personal mobility, WiFi hotspots, Bluetooth-enabled devices, and so on). Likely, this is going to happen in industrial and factory environments as well, where highly automated production systems can get significant benefits from the introduction of most advanced wireless communication techniques. Indeed, several studies have recently proved that some very popular wireless solutions are suitable for the employment in industrial scenarios, too, including real-time communications at the shop-floor [3].

Adopting wireless communications in industrial environments is particularly appealing as, in principle, it avoids (or, at least, reduces) cabling, which turns out to be cumbersome and/or expensive in many real cases. For instance, reliability problems could arise with cables used to connect moving parts, as a consequence of mechanical stress and/or attrition. In the same way, wireless connections between working cells may increase system flexibility, by reducing set up times and costs. However, while a number of standard solutions and components are already available for wired industrial communications, wireless systems are far from being considered well settled for such kinds of applications. It is worth pointing out that, as a matter of fact, the straightforward introduction of highly innovative solutions in industrial and manufacturing systems has neither been simple nor fast. This is due to several reasons, such as for example reliability, efficiency, safety, cost, and security, just to mention a few. As a consequence, the growth of wireless technologies in industrial applications is not being as quick as in other areas, even though they are envisaged to play an essential role in many next-generation automated production systems.

A point most experts agree on is that wireless communications are unlikely to be able to replace completely the wired solutions currently adopted in industrial scenarios, at least in the mid-term. Indeed, it is worth noting that, in most practical applications, this is not considered a real requirement. What is often needed in real-world factory automation systems is the ability to connect a few components to an already deployed wired communication system that cannot be reached (easily and/or reliably) with a cable. The most common example is a sensor mounted on a moving/rotating axis that has to exchange data with a controller attached to a wired segment.

This kind of problems may be easily solved by providing wireless extensions to the existing wired systems. The resulting configurations are "hybrid" networks in which, typically, wireless segments have limited geographic extension (some tens of meters) and connect only few stations (in the order of ten). In such a kind of networks, controllers (e.g., programmable logic controllers [PLCs], computer numeric controls [CNCs], or industrial personal computers [PCs]) are usually located on the wired segment. This means that wireless networks will have to coexist with more traditional wired communication systems for quite a long time. Consequently, the integration of wireless and wired communications in industrial scenarios is becoming a crucial issue that has to be dealt with carefully to achieve both reliability and performance.

This chapter focuses on some of the most relevant aspects concerning the aforementioned hybrid networks. After a general analysis of hybrid wired/wireless configurations and the related issues, the ways wireless extensions can be effectively implemented for both fieldbuses and real-time Ethernet (RTE) networks are described. Some examples are then provided that are based on popular networks: in particular, the well-known Profibus DP [4] and DeviceNet (controller area network [CAN]) [5] fieldbus networks are taken into account, as well as the EtherNet/IP [6] and Profinet IO [7] RTE networks. For the sake of truth, it is worth remembering that many other interesting solutions actually exist that could be provided with a wireless extension. However, as it would have been practically unfeasible to deal with all of them, it has been decided to focus on a reduced set of networks that are nevertheless representative of a wider class of industrial solutions.

Both the IEEE 802.11 wireless local area network (WLAN) [8] and the IEEE 802.15.4 low-rate wireless personal area network (LR–WPAN) [9] have been considered as candidate technologies to implement wireless extensions. WLANs have already been considered in many other studies, and tested in practical applications that demonstrated their suitability for the use in industrial environments

[10–12]. For such a reason, most examples provided in the following are based on that solution. On the other hand, LR–WPAN is an emerging standard for short-range connectivity, mainly conceived for wireless sensor networks (WSNs) that is characterized by long battery lifetime and low cost. Such features allow, for example, sensor networks to be deployed that are distributed over wide geographical areas and can also be used to connect sensors/actuators located on mobile equipment.

Bluetooth [13] is a further, possible candidate for implementing wireless extensions. Such a kind of network, which was originally designed as a mere cable replacement system for mobile equipment, has progressively gained popularity thanks to its features, and is now employed in several areas, including industrial automation. For example, in Ref. [14], a real-time wireless sensor/actuator system based on Bluetooth is described. That system, which uses a modified version of the original medium access control (MAC) protocol, has been practically implemented and compliant components are currently available as commercial products. However, in spite of its interesting features, Bluetooth is not going to be further considered in this chapter for the same page room limits mentioned above.

26.2 Relevant Industrial Networks

In this section, the basic operating principles and the main features of some relevant wireless and wired industrial networks that are considered as possible candidates for setting up hybrid communication systems are briefly described.

26.2.1 Profibus DP and Profinet IO

Profibus and Profinet are industrial networks supported by the PROFIBUS & PROFINET International (PI) global organization. Each one of them includes several communication profiles (CPs) specifically tailored to different application scenarios.

Profibus DP is undoubtedly one of the most widespread fieldbuses. From a technical point of view, it is based on a master–slave scheme which implements data exchanges between controllers ("masters") and field devices such as sensors and actuators ("slaves") at the maximum speed of 12 Mb/s. A further type of communication, namely, master–master, is actually foreseen by the Profibus DP standard. However, this type of communication is typically meant for the non-real-time exchange of diagnostic and/or parameterization data and, for such a reason, it will be no longer considered in the following.

The data-link layer protocol of Profibus DP, known as fieldbus data link (FDL), is based on a token-passing scheme. In particular, a special frame called token, which grants exclusive access to the physical medium, is continuously circulated among the master stations during the network operation. However, it is worth remarking that most of deployed configurations include one master station only (they are commonly referred to as monomaster networks). Slave devices are only allowed to access the network (i.e., to transmit data) when they are queried by a master station.

The operation of Profibus DP relies on a polling cycle that is repeated continuously by the master on slave nodes. During the cyclic operation, the master sends a request telegram to each slave containing the output data. At the same time, it fetches input data made available from the queried device, which are included in the response telegram. Acyclic data may be exchanged as well by means of suitable techniques explicitly foreseen by the Profibus DP protocol.

Profinet IO is an RTE network encompassed by Part 2 of the IEC 61784 International Standard [15], which defines several CPs. In particular, the CP 3/6 refers to both Profinet IO RT_Class 2 and 3 specifications, whereas both CP 3/4 and CP 3/5 are concerned with Profinet IO RT_Class 1. The standard specifies that different types of stations may be connected to the network. Most significant stations are I/O controllers (which are typically powerful devices such as, for example, PLCs or industrial PCs) and I/O devices (sensors and actuators).

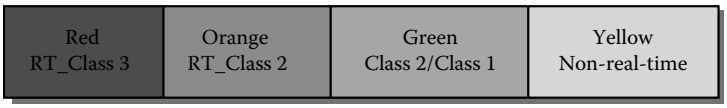


FIGURE 26.1 Profinet IO transmission cycle.

The medium access protocol of Profinet IO is based on a time division multiple access (TDMA) technique. In practice, the traffic is scheduled according to a cycle, which consists of four phases, as shown in Figure 26.1. The first phase (RED) is reserved for RT_CLASS 3 traffic (the most critical) that takes place over predefined physical links enforced by a special kind of switches purposely developed for Profinet IO. In the ORANGE phase, only RT_CLASS 2 frames are exchanged. This is still time-critical traffic; however, in this case, requirements for physical links are not defined. The GREEN phase is reserved for both RT_CLASS 2 and RT_CLASS 1 frames, either cyclic or acyclic. This is the only mandatory phase; the access to the physical medium is regulated by the priority assigned to frames, as specified by the IEEE 802.1Q [16] standard. Finally, the YELLOW phase is reserved for non-real-time traffic.

The application layer protocol of Profinet IO, analogously to other RTE networks encompassed by the IEC 61784 standard, is based on the definition of communication objects that can be exchanged over the network via communication relationships established between I/O controllers and I/O devices.

26.2.2 DeviceNet and EtherNet/IP

DeviceNet and EtherNet/IP (together with ControlNet and CompoNet) are communication solutions that rely on the well-assessed common industrial protocol (CIP) [17]—formerly known as "control and information protocol"—which enables a high degree of interoperability among these kinds of networks. Figure 26.2 depicts the protocol stack foreseen by CIP. CIP can be placed on top of several networks that include (but are not limited to) CAN and Ethernet. Seamless bridging and routing are natively supported by this protocol, which allows parameterization information and process data to be easily exchanged in heterogeneous-distributed systems that rely on both fieldbus and industrial Ethernet transmission technologies.

DeviceNet adopts the CAN protocol at the low end of its protocol stack (physical and data-link layers). Despite the plain bus topology and the relatively low speed (from 125 up to 500 Kb/s), this

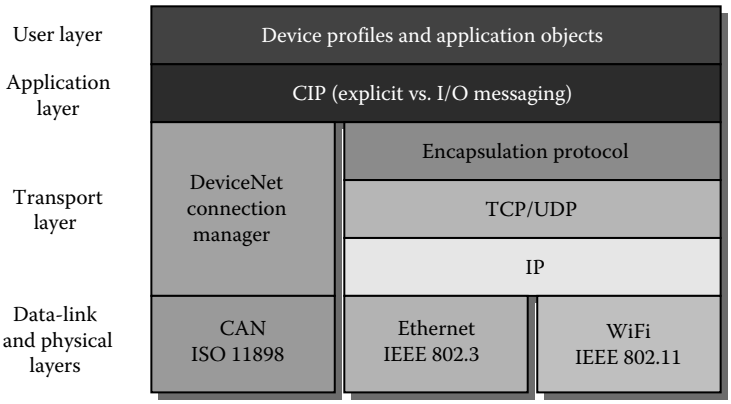


FIGURE 26.2 Protocol stack of CIP-based networks.

solution is able to guarantee deterministic behavior thanks to the bitwise arbitration technique of CAN. This protocol, in fact, basically implements a non-preemptive priority-based communication system, which does not suffer from destructive collisions and where the message characterized by the highest priority is always given precedence for transmission. Besides, providing deterministic behavior and a high level of robustness, this also means that network congestion is implicitly prevented (hence, almost all the theoretical bandwidth is effectively available to control applications). Moreover, CAN is able to feature a good degree of efficiency when small-sized process data are exchanged.

EtherNet/IP is conceived to rely on conventional Ethernet transmissions (theoretically, both half- and full-duplex network can be employed). Although strictly deterministic operations are not ensured, at least from a general viewpoint, the Open DeviceNet Vendor Association (ODVA)—the organization which is in charge of managing CIP—has established a set of design rules [18], which achieve quasi-real-time behavior in EtherNet/IP. Basically, they require that high-speed switched Ethernet is adopted (i.e., full-duplex 100 Mb/s equipment) and that unnecessary network traffic is kept as low as possible. For instance, virtual local area networks (VLANs) or internet group management protocol (IGMP) snooping could/should be used to confine multicast traffic that is generated as a consequence of process data exchanges carried out over I/O connections according to the producer/consumer model. As long as the network load is well below the theoretical available bandwidth, non-blocking switches are able to guarantee that every message is eventually delivered to the intended destination within a period that, because of the high network bandwidth, is short enough for the vast majority of control applications in factory automation systems.

26.2.3 IEEE 802.11

The IEEE 802.11 family includes several standard specifications for WLAN. All devices belonging to such a family (also referred to as WiFi) work in specific bands of the radio spectrum, centered around either 2.54 GHz (legacy 802.11, 802.11b, 802.11g, 802.11e) or 5 GHz (802.11a). High transmission bit rates are foreseen in IEEE 802.11 standards, ranging from 11 Mb/s (802.11b) to 54 Mb/s (802.11g/e/a). Moreover, they are expected to increase further—up to (about) one order of magnitude—with the next-generation devices (which will be based on 802.11n).

The IEEE 802.11 standard specifies a mandatory distributed coordination function (DCF) to access the physical medium that is based on the carrier sense multiple accesses with collision avoidance (CSMA/CA) technique. As a further, non-mandatory, option, IEEE 802.11 includes a centralized MAC protocol as well that is the point coordination function (PCF). In this case, access to the network is regulated by a specific station, namely, the PC that grants exclusive access to one wireless station at a time thus preventing collisions (the primary cause of nondeterminism in WLANs). Although PCF is currently not supported by the vast majority of commercial WiFi boards (with a few noticeable exceptions), it nevertheless represents an interesting option for industrial applications, at least from a theoretical point of view.

Finally, it is worth mentioning that devices complying with the IEEE 802.11e standard also support the concept of Quality-of-Service (QoS), in the form of traffic prioritization and/or parameterization. Despite it was originally conceived for supporting multimedia traffic, such a feature is particularly appealing for industrial applications, too, as it allows critical and urgent messages to be delivered in shorter times. In particular, the IEEE 802.11e specification defines an additional "hybrid coordination function" (HCF), which in turn consists of two distinct mechanisms that are the enhanced distributed channel access (EDCA) and the HCF controlled channel access (HCCA). Despite HCCA is able to provide a fairly more deterministic behavior than EDCA, thanks to the presence of a hybrid coordinator that effectively ensures stations contention-free access to the wireless medium—by allocating them suitable transmission opportunities—compliant devices are, at the time of writing, not available yet. EDCA-compliant equipment, instead, is currently available off-the-shelf at low cost. EDCA is noticeably simpler and cheaper than HCCA, as it basically relies on a completely distributed

approach. Hence, chances are that it will be more and more adopted in the next years as the wireless solution of choice for many real-world industrial plants.

26.2.4 IEEE 802.15.4

The IEEE 802.15.4 standard deals with *ad hoc* wireless interconnections of electronic devices within limited ranges (some tens of meters) at low data rates (up to 250 kb/s). The standard potentially targets several application fields including, for example, home and building automation applications, smart tags communications, cable replacement, and automotive sensing. Devices compliant with the IEEE 802.15.4 standard typically operate in the industrial, scientific, and medical (ISM) band, although a second band is available as well, depending on the country considered, but not widely adopted due to its limited data rate (20 Kb/s).

The IEEE 802.15.4 standard specifies two medium access protocols, namely, the beacon-enabled MAC protocol, characterized by the presence of a "network coordinator," and the non-beacon MAC protocol. In the beacon-enabled mode, the network coordinator periodically issues "superframes" that are divided into two parts. The first part, called contention access period (CAP), takes place immediately after the beacon and is based on a distributed CSMA/CA mechanism that handles the access to the channel. After the CAP, there might be an optional contention free period, in which guaranteed time slots (GTSs) are exclusively allocated to the nodes. Conversely, in the non-beacon mode a fully distributed channel access method is realized by means of a pure CSMA/CA medium access technique.

26.3 Implementation of Hybrid Networks

From a general point of view, different (and often dissimilar) communication networks can be interconnected through specific devices called intermediate systems (ISs) [19]. These devices have different structure, functionality, and complexity, according to the layer of the OSI reference model they refer to. In the following, some practical implementations of these devices are described, along with their features and related issues.

26.3.1 Interconnections at the Physical Layer

The simplest forms of ISs are "repeaters." They operate at the physical layer and are used to interconnect subnetworks that share the same MAC mechanism. Repeaters are usually adopted for regenerating physical signals (electrical, optical, etc.) flowing across different network segments. In Ethernet networks, for example, they operate according to the so-called "3-R principle": re-shaping, re-timing, and re-transmitting. In this way, it is possible to face the problem of signal attenuation over the cable when the network extension grows larger. Moreover, they enable the number of nodes that can be networked—which is practically limited by the current drained by devices attached to the bus—to be increased, and achieve segmented network topologies in addition to the plain linear bus.

Besides simpler two-ported devices, a special kind of repeater exists (called "hub") that is provided with several ports and can be adopted to set up networks based on star topologies. Because of the increased reliability (a defective cable no longer affects the whole network operation, hence avoiding partitioning faults), hubs are particularly suitable for deploying network infrastructures and for the cabling of buildings. More advanced versions of repeaters may sometimes be employed to interconnect subnetworks that rely on different media (e.g., copper wires and fiber optics) or signaling techniques (e.g., voltage/current levels).

When network segments are connected through repeaters, the MAC mechanism that regulates the access to the shared transmission medium is exactly the same on the whole network. Thus, care has

to be taken about the increased propagation delays. In many cases, such as for example in half-duplex Ethernet, CAN, or Profibus networks, this leads to limitations on the maximum number of repeaters that can be inserted between any two nodes or the maximum cable length for every segment.

Pure repeaters are quite unusual in hybrid wired/wireless networks, as the resulting physical communication support in most practical cases is not seen as a sufficiently uniform medium by the MAC protocol. This is because the signaling schemes and physical transmission techniques adopted by wireless networks are, typically, rather different from wired solutions, which means some form of non-trivial buffering has to be carried out by ISs.

An interesting example of hybrid wired/wireless interconnection carried out at the lowest layer of the protocol stack (i.e., below the MAC) is provided in Ref. [20]. In that case, the authors refer to a hybrid architecture between Profibus and Radio Fieldbus [10] that relies on "cut-through" forwarding devices located between wired segments and the wireless medium. Both networks use the same data-link layer protocol—specifically, the FDL as defined by Profibus, including its access scheme based on token-passing—but rely on completely different physical layers. Indeed, Profibus is based on the well-known RS 485 standard, whereas Radio Fieldbus uses a direct sequence spread spectrum technique similar to the one adopted in IEEE 802.11 WLANs.

26.3.2 Interconnections at the Data-Link Layer

An IS operating at the data-link layer is called "bridge." Its main purpose is transferring frames between systems that are not (or cannot be) treated as a uniform communication support by protocols at the MAC level (e.g., switched LANs that are made up of several separate collision domains). Such devices usually operate according to the "store & forward" principle, that is, when a frame is completely received on one port of the bridge, it is forwarded (possibly, according to a selective strategy based on adaptive message filtering) to the other subnetwork(s) through the related port(s). As an alternative, forwarding may start as soon as a sufficient amount of information is received about the incoming frame (e.g., cut-through bridges). Even though featuring shorter transmission latencies, in this latter case erroneous frames are not blocked by the bridge, hence reducing the net available bandwidth.

It is worth noting that, thanks to this decoupling mechanism, physical signaling, transmission speeds, medium access techniques, and even the frame formats might differ in the interconnected networks. A bridge, which is equipped with more than 2 ports, is commonly referred to as a "switch" (this kind of device is currently present in almost every real-life Ethernet network).

Even though the MAC mechanism is not required to be the same for the subnetworks interconnected through bridges/switches, the sets of communication services provided at the data-link layer should be at least similar. Limitations on the kinds of networks that can be interconnected concern, for example, their addressing scheme (which has to be uniform on the whole network) and the maximum transfer unit (MTU) that affects the allowed payload size (this is because fragmentation is not permitted at the data-link level). When subnetworks with different MTUs are interconnected, care has to be taken so that the payload of the exchanged frames never exceeds a threshold equal to the smallest among the supported MTUs.

Figure 26.3 shows an example of interconnection taking place through a bridge for the hybrid networks considered in this chapter. From a practical point of view, the bridge used in hybrid wired/wireless networks is a device equipped with two (or more) transmitting/receiving interfaces (one for each subnetwork) and, optionally, a protocol converter. A frame originated from whatever segment (either the wired network or the wireless extension) is propagated to the other one by the bridge, which receives the frame, converts it into the suitable (target) format, and then retransmits it.

When real-life devices and solutions are considered, access points (APs) are a very popular example of bridge. They are used to set up hybrid networks consisting of one (or more) wired IEEE 802.3 segments and an IEEE 802.11 wireless extension. Data-link layer services, in this case, are exactly the same for the two kinds of networks (i.e., those foreseen for the IEEE 802 family of protocols),

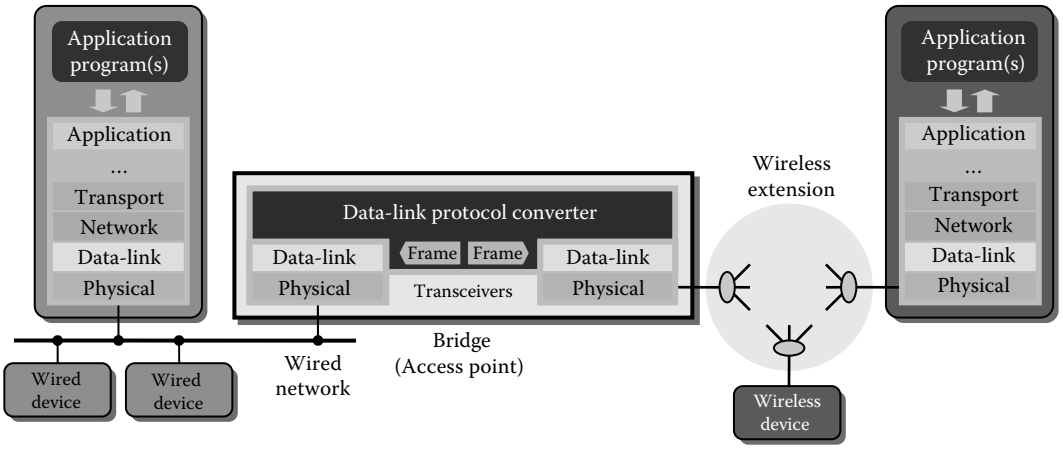


FIGURE 26.3 Interconnection of networks via a bridge.

as well as the 6-byte addressing scheme. On the contrary, frame formats and MTUs differ. For the sake of truth, it is worth remembering that APs actually have a twofold role: they are used both to enable communication in an infrastructure WiFi network (also known as basic service set [BSS]), by relaying frames among wireless stations, and to interconnect the BSS to an existing Ethernet backbone ("portal" function).

26.3.3 Interconnections at Higher Protocol Layers

For ISs working at the network layer or above, the generic term "gateway" is often adopted. A gateway is responsible for transferring protocol data units (PDUs), as well as for converting generic streams of information and application services between application processes executing in the nodes of the interconnected systems, by performing all required protocol translations. From a practical point of view, there is no particular restriction on the kinds of networks that can be interconnected through gateways. On the other hand, gateways are usually complex, expensive devices, often developed *ad hoc* for the considered systems by means of suitable software modules, and the performance they achieve are to several extents lower than bridges. Generally speaking, noticeably higher latencies have to be expected in this case, which could result incompatible with timings requirements of control applications.

ISs that operate specifically at the network layer are known as "routers." At this level, a uniform addressing scheme and protocol are used (i.e., the Internet protocol [IP]), which ensures worldwide connectivity irrespective of the physical media, MAC mechanisms, and data-link services of the interconnected subnetworks. In this chapter, routers are not specifically addressed, as most industrial networks are deployed as local networks. However, when timing constraints are not very tight, interconnection techniques that rely on routers allow devices, which are natively enabled to communicate over local intranets and/or the Internet, to be directly embedded in the control system (this is the case of remote monitoring and maintenance tasks).

When the interconnection takes place at the application layer, the IS should take care of the conversion of high-level services. Usually, this implies the gateway has to carry out non-trivial operations for gathering and converting information which, in turn, often requires a number of simpler communication services to be invoked on the interconnected networks. In these cases, the term "proxy" is sometimes adopted. Indeed, a proxy is much more than an application-layer gateway: it usually makes some parts of the network (e.g., a whole subnetwork) look like as if they were a single node, possibly hiding the underlying structure for security or performance reasons. Sometimes, some form

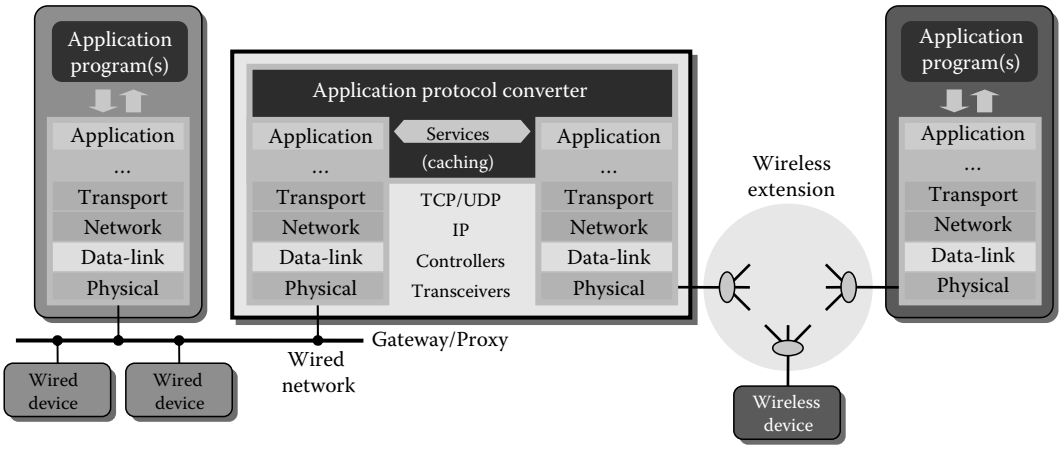


FIGURE 26.4 Interconnection of networks via a gateway.

of data caching is provided as well to save bandwidth. For instance, proxies are used in fieldbus/IP configurations to connect legacy fieldbus segments. In this case, they integrate devices connected to the underlying fieldbus into the overlying Ethernet system. In this way, several advantages of fieldbus networks, such as the high responsiveness, powerful diagnostic functions, and automatic configuration can be maintained, and at the same time planning is simplified (because workflows are well-known) as well as commissioning and operation (thanks the diagnostic capabilities offered by the fieldbus).

An example of wireless extension implemented at the application layer is shown in Figure 26.4. The gateway in the picture includes two interface boards that are fully compliant with the overall protocol stack of the relevant subnetworks, and a protocol converter, usually implemented as a software module.

26.3.4 Wireless Extensions of Industrial Networks

Although technically feasible, wireless extensions of industrial networks to be used at the shop-floor (either fieldbuses or RTE networks) are not so straightforward. This is basically due to three main reasons: first, the transmission support (wireless medium) is shared among all nodes (possibly including those associated to nearby independent wireless subnetworks). Even operating at the maximum allowable speed (e.g., 54 Mb/s for the currently available IEEE 802.11a/g/e compliant networks), this may result in a low per-station net throughput—compared to that achieved in many wired networks—when the number of connected devices grows higher (this is particularly true when either switched Ethernet or industrial Ethernet solutions based on a combined message are taken into account). Furthermore, the nonnegligible overhead introduced by the communication protocol (larger protocol control fields, acknowledgment, and reservation mechanisms, no provision for full-duplex operations, and so on) has to be considered as well. For example, the minimum time needed to reliably exchange 8 bytes of user data over a WLAN—i.e., bit rate equal to 54 Mb/s, no RTS/CTS mechanism, no TCP/IP encapsulation, *ad hoc* network mode, but including interframe spaces as they effectively waste network bandwidth—is about 100 μs. Such a value is only slightly shorter than the worst-case time taken to send the same information over a CAN network operating at 1 Mb/s (i.e., 135 μs).

The second reason is that random medium access techniques (e.g., CSMA/CA) are often employed by wireless networks. This means that, on the one hand, unpredictable—and unbounded—transmission delays might occur (because of the occurrence of repeated collisions) while, on the other

hand, network congestions could be experienced, and this is surely the worst aspect. In other words, when the network load generated by devices exceeds a given threshold (even for a limited time), a condition may likely arise—because of the positive feedback due to the increased collision rate—so that the effective net throughput decreases as the load increases. This means that the network may become temporarily unavailable to carry out data exchanges timely, which is not compatible with proper operation of distributed control systems. The number of lost messages (discarded by the sender because the retry limit is exceeded) may even grow up to a point no longer acceptable for the control application. Moreover, even in the case of lightly loaded networks, non-negligible jitters may affect data exchanges and hence the accuracy of the system, unless some form of prioritization scheme is adopted for frame transmission.

Third and last, wireless channels are much more error-prone than wired cabling [21], and this is a serious drawback when they are used in those industrial environments that are often plagued by non-negligible electromagnetic interferences. Besides causing higher communication latencies and jitters, transmission errors directly affect the network reliability and, consequently, the robustness of the overall system. They may also cause consistency problems—for instance, when a multicast message is received only by a part of the addressed devices.

Unless acknowledged transmission schemes are adopted (which, incidentally, are not allowed when broadcast and producer/consumer-like multicast traffic is taken into account), there is a non-negligible chance that messages sent over the wireless medium never reach the intended destination, even in non-noisy environments. Unfortunately, if electromagnetic disturbance is anything but low (as in real plants) acknowledgments are no longer sufficient to ensure reliable connections.

Each one of the three drawbacks mentioned above can be tackled (at least partially) by means of already existing or soon available technologies. In particular, the throughput problem could be somehow lessened by the upcoming IEEE 802.11n specification, which is based on the multiple-input multiple-output technology and is theoretically able to provide a big leap (about one order of magnitude) in the network throughput. Indeed, as process data are usually small-sized, the same level of improvement cannot be reasonably expected in industrial scenarios, where simple field devices are interconnected. A more viable choice is relying on several (smaller) separate wireless subnetworks (e.g., WLANs), possibly operating on different channels, interconnected by means of a wired backbone to limit the packet rate on each one of them. Keeping the size of each wireless subnetwork small also helps high speed operation (because, in this case, the automatic rate adaptation mechanism featured by most APs and WLAN adapters is not activated).

The second problem (determinism) can be tackled through the adoption of enhanced medium access techniques, such as PCF in the case of IEEE 802.11. Unfortunately, PCF is seldom implemented in commercially available APs. Variants of PCF, such as the iPCF mechanism introduced by Siemens [22], have the drawback of being proprietary solutions, and hence compatibility and interoperability with standard equipment can hardly be ensured. Alternatively, the new prioritization features offered by the EDCA function of the IEEE 802.11e standard (already supported by many WLAN adapters currently available off-the-shelf) might be exploited [23]. While not guaranteeing strict determinism, such a technology (mainly developed for multimedia applications) is able to provide tangible improvements, such as, for example, guaranteeing shorter transmission times "on the average" for urgent notifications, which might be often enough for several industrial automation systems. For example, it is possible to ensure quasi-real-time behavior by assigning higher priorities (i.e., voice and video access classes) to those messages characterized by tight timing/safety constraints.

Also TDMA techniques may be useful to solve the determinism problem. In such a case, for example, the GTSs provided by IEEE 802.15.4 represent an interesting opportunity. Other solutions, recently appeared in literature [24], suggest the adoption of synchronization techniques on WLANs, to reduce the likelihood of a collision for data exchanges that take place according to predictable (e.g., periodic) patterns.

The third problem (robustness) can be faced in several ways. Whenever feasible, a proper selection and placement of antennas is envisaged, possibly by exploiting antenna diversity. Purchasing APs and adapters with removable antennas is often a good idea, as this permits boosting the strength of signals by replacing the built-in omnidirectional antenna with a high-gain or directional one. Moreover, wireless range extenders can be used to deal with the problem of dead spots (areas with no or weak reception)—even though setting up such devices in a proper way is not always straightforward.

In the case this is not enough, leaky wave antennas [25] can be used to ensure reliable communication over radio links. The drawback, in this case, is the need to deploy the leaky cable infrastructure, which makes it a non-universal solution. Usually, such an arrangement is adopted when the equipment includes some parts moving on fixed paths, to avoid cable wear and strain. As an alternative, if true mobility is required, devices operating in the 5 GHz band (IEEE 802.11a) can be adopted as a satisfactory choice [21]. In this way, in fact, interferences with other devices (mobile phones, Bluetooth-enabled equipment, notebooks with wireless connections, and so on) operating in the standard (and already jammed) 2.4 GHz ISM band are avoided, hence achieving a better signal-to-noise ratio and, consequently, a reduced error rate on the radio link.

26.4 IEEE 802.11-Based Extensions: Fieldbuses

Typically, the transmission protocols adopted by fieldbuses are noticeably different from those employed in wireless networks, at all levels of the communication stack. For example, to the best of our knowledge, random access techniques typical of WLANs are not employed in any fieldbus network (they are sometimes used in building automation networks, which are characterized by relaxed timing constraints). Other problems may arise concerning either the payload size of frames or the address space, both of which are typically quite small in real-world fieldbuses (few tens/hundreds of bytes/nodes, respectively).

As a consequence, wireless extensions of fieldbuses have to be implemented mainly at the application layer, that is, through gateways. However, some remarkable exceptions exist: DeviceNet, for instance, ensures interoperability with all networks based on the CIP protocol, thanks to its native routing techniques that allow, in principle, the implementation of extensions operating below the application layer (i.e., through direct routing of CIP messages).

26.4.1 Extensions of Profibus DP

A gateway able to extend Profibus DP may be directly implemented on a master station as shown in Figure 26.5. The master has to be equipped with an IEEE 802.11 board and suitable code has to

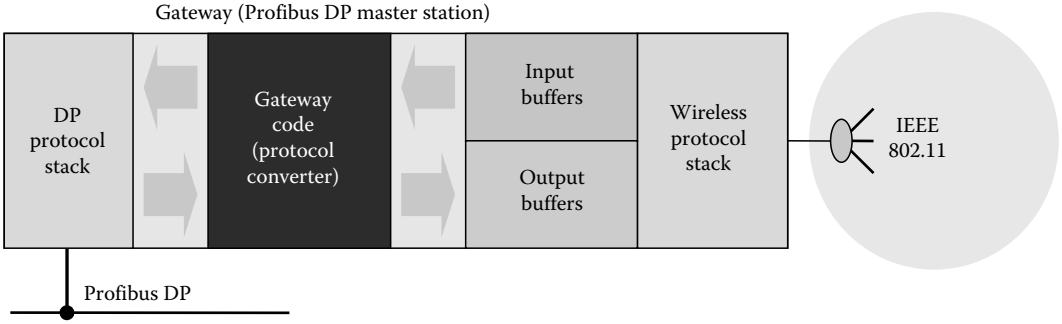


FIGURE 26.5 Gateway for Profibus DP with proxy functionality.

be developed, in this case, to enable the exchange of data between the two segments. Wireless nodes belonging to the extension are organized in a BSS, which represents the building block of a WLAN. A similar solution was proposed in Ref. [11] for the previous version of Profibus, namely, Profibus FMS. The practical application described in such a paper allowed to achieve some interesting results derived from a set of experiments. In particular, the stations of the BSS were queried according to a virtual polling algorithm; each station exchanged 40-byte with the gateway, resulting in a polling time for wireless stations as low as 5 ms.

Clearly, the use of a gateway introduces additional delays. However, their effects may be considerably lessened by the adoption of a proxy. Such a device maintains an image of the process input/output data (decoupled from data exchanges, which take place actually on wireless segments) accessible from the wired segment at any time, without restrictions introduced by the wireless communication protocol. The example in Figure 26.5 shows that this may be accomplished by means of a set of input/output buffers. Stations on the wireless segment access these buffers to store input data acquired from the controlled process. Similarly, output data stored into the proxy output buffers may be subsequently retrieved by wireless stations.

The proxy code allows interfacing the Profibus DP protocol to the input/output buffers. Specifically, when a data exchange request, that carries output data, is issued by the Profibus DP master protocol, the proxy extracts such data from the PDU and writes them into the related output buffers. At the same time, input data are picked up from the input buffers and encapsulated in the acknowledgment frame, which is returned to the Profibus DP master in accordance with the protocol rules. It is worth noting that the above technique does not impose any specific restriction on the protocols actually used on both the wired segment and the wireless extension. Indeed, data exchanges take place via input/output buffers so that they are effectively decoupled. For example, the virtual polling algorithm described in Ref. [11] could be replaced by an implicit token-passing technique, which does not require specific queries of the wireless nodes.

26.4.2 Extensions of DeviceNet

DeviceNet was designed bearing in mind the option of having several subnetworks interconnected by means of routers. This mechanism was conceived initially to overcome the drawbacks of the CAN protocol, which limit every DeviceNet segment (that coincides with one arbitration domain) to 64 nodes at most and no more than few hundred meter extension. However, it proves to be useful for interconnecting other kinds of networks as well. Seamless routing capabilities set DeviceNet apart from most of the other fieldbus solutions. In particular, a set of suitable objects have been included in the specification that defines the mechanisms a router can use to forward messages from one network port to another.

Besides making the interconnection with other ODVA networks (e.g., ControlNet and EtherNet/IP) simpler, such a feature turns out to be helpful when implementing wireless extensions. Routing in heterogeneous CIP networks is achieved by means of the so called "port segments," which are added to messages and explicitly describe the route they should follow (this is often referred to as "source routing"). The route is specified through a list of items. Each item, in turn, describes a "hop" by means of two pieces of information: the output port of the router used to forward the message and the address of the intended destination in the next subnetwork (that can be either the target node or another router). As the frame travels along its route, the port segment is shortened by routers: in particular, each one of them removes the item concerning the hop it has processed. Obviously, the routing process takes some time, and hence response times over the interconnected network increase. This aspect has to be taken into account carefully when control applications with demanding real-time requirements have to be designed (for example, timing constraints might need to be relaxed).

Using WLANs together with conventional DeviceNet devices can be accomplished in several ways. The first, simpler, solution is to rely on an existing DeviceNet to EtherNet/IP router that, in turn, is

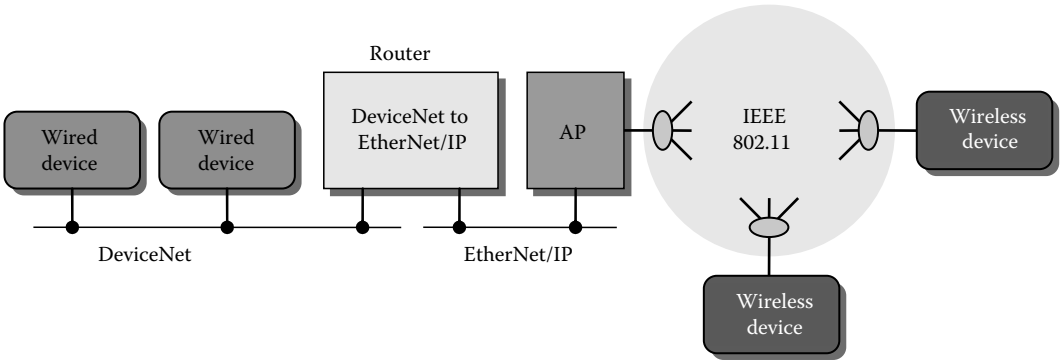


FIGURE 26.6 Extension of DeviceNet via a CIP router and an AP.

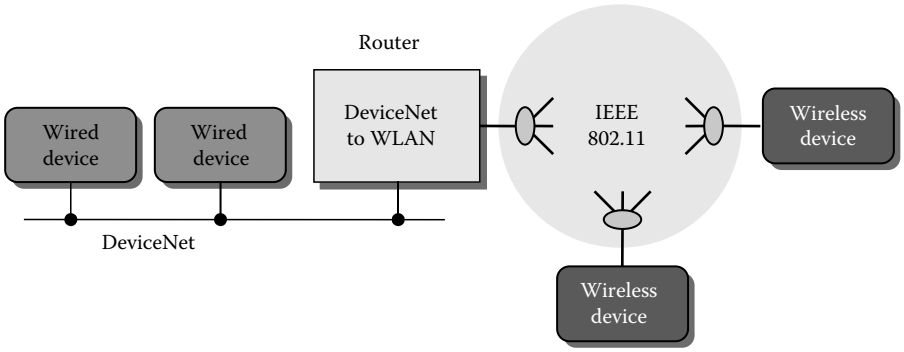


FIGURE 26.7 Extension of DeviceNet via a WLAN-enabled CIP router.

connected to a conventional AP (see Figure 26.6). In this case, no custom hardware component has to be developed. However, this solution causes each message to traverse (at least) two intermediate devices before reaching its target node (e.g., the router and the AP), and this introduces additional delays.

A second and more sophisticated solution requires the existing DeviceNet routers to be modified, to include an IEEE 802.11 port too, as shown in Figure 26.7 (likely, this should not be a difficult task for manufacturers of DeviceNet products). In this case, as the wireless-enabled router is completely aware of the exact requirements of each CIP connection (including details about its transport class and expected packet rate), it can select the most suitable QoS when forwarding messages over the radio channel. If a QoS-enabled network is adopted on the wireless side, such as for example IEEE 802.11e, an improved real-time behavior could be achieved.

It is worth noting that, at present, a couple of devices are available off-the-shelf that can be used to implement wireless extensions to DeviceNet networks. For instance, the solution described in Ref. [26] relies on message routing to accomplish information exchanges over explicit connections. In this case, a "wireless master" (implemented as a DeviceNet slave) is connected to the primary DeviceNet bus, whereas one or more "wireless slaves" are used to connect separate DeviceNet remote buses (every wireless slave is seen as the virtual DeviceNet master of the relevant remote bus). To improve communication efficiency, a mechanism is also foreseen that makes wireless slaves collect process data from the attached remote buses and provide them to the master as a single process image. Unfortunately, solutions like this one are mainly based on proprietary wireless communication technologies, and hence they can hardly be viewed as universal proposals.

26.5 IEEE 802.11-Based Extensions: RTE Networks

Despite the wide availability of both hardware components and protocols, the implementation of wireless IEEE 802.11-based extensions to RTE networks is not as straightforward as one could think. Indeed, interoperability between Ethernet and WLANs is always possible by means of inexpensive off-the-shelf devices (i.e., APs), along with some commonly available protocols like the well-known IEEE 802.1D bridging protocol [27], which allows for interconnections at the data-link layer. Despite the good efficiency, implementing this kind of solutions in industrial environments might be sometimes not so easy, due to access protocols adopted by RTE networks, which are often not entirely compliant with the original Ethernet specifications. Moreover, the unavoidable uncertainty affecting the IEEE 802.11 communication caused by the random back off procedure, interferences, and electromagnetic noise may seriously impair the determinism of the RTE networks. An example of IEEE 802.11-based wireless extension is provided in Ref. [28] for a network different from those considered in this chapter. There, the authors refer to Ethernet Powerlink (EPL) [29], a very popular RTE network that adopts an access protocol based on a fast and precise periodic polling carried out on the (wired) nodes; this ensures fast operation (cycle times under 200 μs) and ultralow jitter (below 1 μs). Results presented in the paper actually confirm the impossibility of maintaining the aforementioned tight timing performance when wireless nodes are introduced. Nonetheless, the adoption of prioritized frames made possible by IEEE 802.11e revealed to be quite interesting as it allows for a consistent reduction of jitters, especially when a limited number of wireless stations are employed.

A more straightforward interconnection technique, in general, can be obtained through the communication services offered by the TCP/IP protocol suite. Unfortunately, given the complexity of TCP, this solution is not suitable for achieving real-time behavior. For such a reason, in several cases the UDP protocol [30] is preferred. UDP, in fact, has lower overheads than TCP and provides simpler communication services, thanks to its unacknowledged transmission scheme (which lacks of the sliding window mechanism of TCP). This implies that latencies and jitters become shorter than in TCP, even though the overall behavior cannot be considered as strictly deterministic, yet.

26.5.1 Extensions of Profinet IO

The extension of Profinet IO has to face all issues discussed in the previous sections. In particular, an extension at the data-link layer cannot be implemented, as the CSMA/CA technique employed by IEEE 802.11 cannot cope with the very tight time schedule imposed by the TDMA medium access technique used by Profinet IO, even if prioritized frames are used. As a consequence, the extension may only be implemented at the application layer, via a gateway, similarly to that described for Profibus DP.

The structure of Profinet IO allows for the implementation of wireless extensions in two different ways. When a real-time behavior is not required, the data exchange with the wireless nodes may take place in the non-real-time period of the Profinet IO cycle (the YELLOW phase shown in Figure 26.1). The second option is more performing, as real-time properties can be preserved. Figure 26.8 shows that an AP implementing the PCF technique can be used as a bridge toward the wireless segment. In this way, wireless stations can be directly included in the Profinet IO cycle, as the AP may assign them specific time slots that provide contention-free, deterministic, access to the wireless medium. However, specific actions are needed, in this case, to improve the network reliability (for example, the adoption of leaky cables).

An example of application based on this option is provided in Ref. [31]. In that paper, 21 Profinet IO devices are located on the wireless segment, whereas the Profinet IO controller relies on wired connections. The bridge is a commercial product [22] implementing the iPCF technique mentioned in Section 26.3.4.

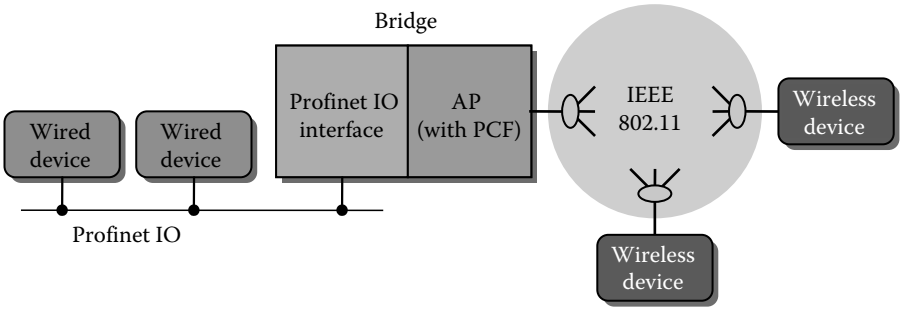


FIGURE 26.8 Extension of Profinet IO via a bridge.

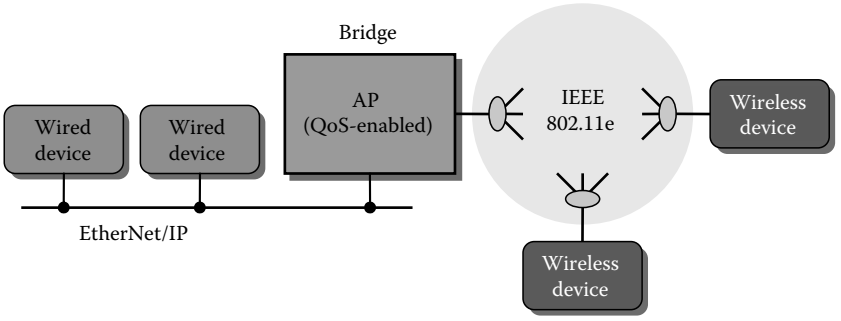


FIGURE 26.9 Extension of EtherNet/IP via an AP.

26.5.2 Extensions of EtherNet/IP

As long as EtherNet/IP networks [6] are taken into account, there is no need for any change to adapt the CIP protocol to IEEE 802.11 networks, at least from a theoretical point of view. APs can be used to forward each message to the intended destination(s) by means of its Ethernet address (which, in turn, is derived from the IP address used at the application level), as shown in Figure 26.9. In fact, both Ethernet and WiFi networks share the same address space, which means that transparent bridging at the data-link layer can be easily achieved. The only issue concerns the MTU on these networks, as the maximum payload allowed on WLANs is greater than that of Ethernet. In particular, the size of the frames sent by stations on wireless extensions has to be limited to the maximum value achieved on the wired segment.

While this is certainly the most appealing (and inexpensive) solution, it might lead to sub-optimal results when real-time process data have to be exchanged (i.e., for I/O connections). This is because, in the existing IEEE 802.11b/g WLANs, differentiated services cannot be offered. In other words, priorities cannot be assigned to process data higher than those envisaged for background traffic, unless special techniques, such as, for example, those provided by IEEE 802.11e, are adopted.

In the latter case, however, existing implementations of CIP have likely to be modified, to map the timing constraints of each message (I/O vs. explicit connection, expected packet rate, and so on) that are known at the application level, onto the most suitable QoS as provided by the underlying wireless communication system. From this point of view, Ethernet is not the same as IEEE 802.11e WLANs. In fact, while IEEE 802.1Q [16] foresees up to 8 "traffic classes" to encode frame priority in switched Ethernet networks, IEEE 802.11e defines four different "access categories" (AC) to provide traffic prioritization. Moreover, while strict priority order is enforced among frames by switches (i.e., the highest priority message is selected for transmission on a given output port among all the queued

frames), EDCA only provides a mean to offer higher-priority access classes an increased likelihood to win the contention (by decreasing the related back off times).

It is worth noting that developing WLAN-enabled EtherNet/IP devices, such as the target node on the right side in Figure 26.9, is not really a difficult task. The advantage of adopting this kind of solution is that devices can exploit the knowledge about the timing requirements of every CIP connection to select the most suitable QoS. A viable choice might be, for example, to map I/O connections with tight timing requirements (i.e., high expected packet rate) on voice-grade (AC_VO) traffic, whereas those having less severe constraints on transmission times can be mapped on video-grade (AC_VI) transmissions. Finally, explicit connections devoted to parameterization activities may rely on either best-effort (AC_BE) or background (AC_BK) traffic.

In the case conventional APs are used to interconnect wired EtherNet/IP segments with wireless extensions, exploiting the IEEE 802.11e QoS property is a non-trivial task. Indeed, a conventional AP cannot determine the exact timing requirements of each frame, as this information is known at the application layer, whereas the AP operates at the data-link layer. A possible solution is to rely on APs that are able to map IEEE 802.1Q traffic classes directly onto IEEE 802.11e ACs (and vice versa). In this way, the hybrid communication system does its best in trying to fit the timing requirements of the different connections.

Besides traffic prioritization, another important issue concerns mechanisms for broadcast/multicast traffic confinement. This is needed as process data, in modern industrial communication systems, are often exchanged according to the producer/consumer paradigm. In wired networks such as EtherNet/IP, traffic confinement is dealt with by means of IGMP snooping or VLANs. Although the latter mechanism is not available in IEEE 802.11 wireless networks, some devices are currently available off-the-shelf that are able to map VLANs to different service set identifiers [32]. Such a feature is quite interesting, as it can be used to ease the integration of WLAN extensions in the existing EtherNet/IP networks.

26.6 IEEE 802.15.4-Based Extensions

The IEEE 802.15.4 specification, as all networks defined by the IEEE 802 committee, is only concerned with the two lowest layers of the communication stack and does not specify any type of upper protocol stack. Thus, the access to IEEE 802.15.4 networks by higher layers may be achieved in two different alternative ways, as recommended by the standard, that is:

1. Via logical link control (LLC) Type 1 services [33]
2. Directly, by means of the MAC services

Both techniques are effective, even if it is worth mentioning that IEEE 802.15.4 boards manufacturers rarely include LLC services in their products.

IEEE 802.15.4 has a low transmission speed and, hence, wireless extensions are not able to provide fast communications in this case. Nonetheless, the high efficiency of the protocol makes it particularly appealing for applications where limited amount of data have to be transmitted. Moreover, the GTSs available in the beacon-enabled mode allow for deterministic access to the shared medium by wireless nodes, with a consequent reduction of the uncertainty typical of CSMA/CA-based protocols.

26.6.1 Extensions of Fieldbuses

The wireless extension of fieldbuses by means of IEEE 802.15.4 suffers from the same problems already discussed for IEEE 802.11. Specifically, differences between the MAC protocols of the two kinds of networks discourage interconnections at the lower layers. Hence, wireless fieldbus extensions based on IEEE 802.15.4 networks can only be achieved through a gateway that maps services foreseen

by the fieldbus application layer onto calls to IEEE 802.15.4 services. The gateway may be typically implemented as a station that embeds the interface to the fieldbus and, at the same time, acts as a coordinator for the IEEE 802.15.4 network.

Also in this case, the implementation of the gateway does not impose any restriction on the protocol(s) actually employed in the wireless extension. In particular, the gateway may rely on functions provided by a proprietary protocol explicitly defined for managing data exchanges over the wireless extension. Alternatively, IEEE 802.15.4 data-link layer services (LLC or MAC) can be directly employed by the gateway to access wireless stations. Once again, it is worth remarking that significant benefits can be obtained from the use of GTSs. As an example, the virtual polling algorithm described in Ref. [11] for IEEE 802.11 does not need to be implemented for IEEE 802.15.4 as, in practice, such a mechanism is natively provided by the beacon-enabled mode of the access technique.

26.6.2 Extensions of RTE Networks

The availability of the LLC protocol for IEEE 802.15.4 networks could enable, in principle, the implementation of extensions to RTE networks right at the data-link layer, as most networks provide access to such a layer as well. However, as Type 1 LLC services are non-confirmed, connectionless services, there is a non-negligible probability of errors affecting communications in the wireless segment. Indeed, IEEE 802.15.4 does not implement any frame retransmission technique at the MAC level. Hence, the recovery action, which is necessary in case of frame corruption, is delegated to upper layers. Unfortunately, the protocol is not able to detect transmission errors with the consequent definitive loss of PDUs. As a conclusion, also in this case, effective extensions can only take place via a gateway.

As a final remark, it is worth mentioning that ZigBee [34] offers a complete protocol stack based on IEEE 802.15.4 and, consequently, it represents an appealing opportunity for implementing wireless extensions of both fieldbuses and RTE networks considered in this chapter. Particularly, ZigBee offers connectivity with whatever type of networks based on the IP protocol, by means of the purposely defined ZigBee expansion devices. Consequently, it can be directly interfaced to several RTE networks (all those that allow for TCP/IP communications such as, for example, EtherNet/IP and Profinet IO) using a conventional IP router.

26.7 Conclusions

Because of the need to provide flexible, inexpensive, and reliable communications in production plants, it is currently envisaged that hybrid configurations resulting from wireless extensions of conventional (i.e., wired) industrial networks will be adopted more and more in the next future. This is mainly possible thanks to the availability of wireless technologies and devices able to cope with industrial communication requirements in a proper way.

Concerning wired networks, the analysis presented in this chapter has focused on four popular solutions, namely, two fieldbuses (Profibus DP and DeviceNet) and two RTE networks (Profinet IO and EtherNet/IP). In the same way, both the IEEE 802.11 family of WLAN standards and IEEE 802.15.4 WSN have been taken into account as wireless extensions, as they are some of the most promising technologies for use in industrial automation and control applications, and a lot of devices are already available off-the-shelf at (relatively) low cost.

As pointed out in the previous sections, the extension of Profibus DP and, in general, of almost all fieldbuses can typically take place at the application layer, through the use of suitable proxies. While providing greater flexibility, this may worsen the overall performance, due to delays this kind of devices unavoidably introduces. As an interesting exception, DeviceNet can be interconnected at

a lower level via CIP routers. This means that in hybrid networks based on DeviceNet all devices (both wired and wireless) can be seen and accessed as conventional CIP nodes, whereas in the case a gateway/proxy is adopted (as it happens in Profibus DP) wireless nodes are no longer effectively part of the Profibus network.

When real-time (industrial) Ethernet networks have to be provided with a wireless extension, more alternatives are possible, in principle, including carrying out interconnection at the data-link layer. However, as these networks often rely on peculiar protocols to ensure predictable communications over conventional Ethernet, the integration with WLANs is not usually straightforward. As a consequence, a loss of performance and determinism in hybrid networks is expected to occur also in this case. Moreover, when IEEE 802.11 is selected for the wireless extension, the native randomness of the DCF medium access technique has to be taken into account carefully in designing/deploying the hybrid network, as it likely leads to additional (and unwanted) delays. Significant improvements can be achieved by using deterministic techniques, such as, for example, PCF in WLANs (which, however, is not so frequently available in real devices) or GTSs in IEEE 802.15.4 networks (that, however, suffer from a noticeably lower bit rate).

Finally, an interesting option for achieving quasi-real-time communications over wireless extensions is provided by the IEEE 802.11e standard, which supports the concept of QoS. It has been shown that, for example, traffic prioritization could be used to deal with different types of connections (and, hence, of timing requirements) defined and handled by the CIP protocol, to transparently enable control applications to meet real-time constraints, even when hybrid network configurations are adopted.

References

1. J. P. Thomesse, Fieldbus technology in industrial automation, *Proceedings of the IEEE*, 93(6), 1073–1101, June 2005.
2. J. D. Decotignie, Ethernet-based real-time and industrial communications, *Proceedings of the IEEE*, 93(6), 1102–1117, June 2005.
3. A. Willig, K. Matheus, and A. Wolisz, Wireless technology in industrial networks, *Proceedings of the IEEE*, 93(6), 1130–1150, June 2005.
4. Deutsches Institut fuer Normung, *Profibus-DP Standard*, Translation of the German National Standard DIN 19245 Part 3, Beuth Verlag GmbH Burggrafenstraße, 6, D-100 Berlin, 30, Germany, 1994.
5. International Electrotechnical Commission, IEC 62026-3: Low-voltage switchgear and controlgear—Controller-device interfaces (CDIs)—Part 3: DeviceNet, July 2000.
6. Open DeviceNet Vendor Association, *CIP Networks Library, Vol. 2: EtherNet/IP Adaptation of CIP*, 1.3 edn, ODVA, Ann Arbor, MI, 2007.
7. PROFIBUS International, *PROFINET IO Application Layer Service Definition*, *Application Layer Protocol Specification*, Version 1.0, March 2004.
8. Institute of Electrical and Electronics Engineers, IEEE Standard for information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: *Wireless LAN Medium Access Control* (*MAC*) *and Physical Layer* (*PHY*) *Specifications*, IEEE Std. 802.11–2007, June 12, 2007.
9. Institute of Electrical and Electronics Engineers, IEEE Standard for information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 15.4: *Wireless Medium Access Control* (*MAC*) *and Physical Layer* (*PHY*) *Specifications for Low-Rate Wireless Personal Area Networks* (*WPANs*), IEEE Std. 802.15.4–2006, September 8, 2006.

10. L. Rauchhaupt, System and device architecture of a radio based fieldbus—The Rfieldbus System, in *Proceedings of the IEEE WFCS 2002*, Vasteras, Sweden, September 2002.
11. S. Lee, K. C. Lee, M. H. Lee, and F. Harashima, Integration of mobile vehicles for automated material handling using Profibus and IEEE 802.11 networks, *IEEE Transaction on Industrial Electronics*, 49(3), 693–701, June 2002.
12. A. Willig, M. Kubisch, C. Hoene, and A. Wolisz, Measurements of a wireless link in an industrial environment using an IEEE 802.11-compliant physical layer, *IEEE Transactions on Industrial Electronics*, 49(6), 1265–1282, December 2002.
13. Bluetooth Special Interest Group. Available [online]: www.bluetooth.com.
14. D. Dzung, J. Endresen, C. Apneseth, and J. -E. Frey, Design and implementation of a real-time wireless sensor/actuator communication system, in *Proceedings of the IEEE ETFA 2005*, Catania, Italy, pp. 433–442, September 2005.
15. International Electrotechnical Commission, IEC61784-2: Industrial communication networks—Part 2: Additional fieldbus profiles for real-time networks based on ISO/IEC 8802-3, IEC Standard, November 2007.
16. Institute of Electrical and Electronics Engineers, IEEE standards for local and metropolitan area networks virtual bridged local area networks, IEEE Std. 802.1Q-2005, 2006.
17. Open DeviceNet Vendor Association, *CIP Networks Library*, *Vol. 1*: *Common Industrial Protocol* (*CIP*) *Specification*, 3.2 edn, ODVA, Ann Arbor, MI, 2006.
18. Open DeviceNet Vendor Association, Network Infrastructure for EtherNet/IP: Introduction and Considerations, Publication Number: PUB00035R0. Available [online]: http://www.odva.org/.
19. F. Halsall, *Data Communications*, *Computer Networks and Open Systems*, Addison Wesley Publishing Company, Redwood City, CA, 1996.
20. C. Koulamas, S. Koubias, and G. Papadopoulos, Using cut-through forwarding to retain the real-time properties of profibus over hybrid wired/wireless architectures, *IEEE Transactions on Industrial Electronics*, 51(6), 1208–1217, December 2004.
21. D. Brevi, D. Mazzocchi, R. Scopigno, A. Bonivento, R. Calcagno, and F. Rusinà, A methodology for the analysis of 802.11a links for safety-critical industrial communications, in *Proceedings of the IEEE WFCS 2006*, Torino, Italy, pp. 165–174, June 2006.
22. Siemens AG, SCALANCE W788 RR Access Point.
23. G. Cena, I. Cibrario Bertolotti, A. Valenzano, and C. Zunino, Evaluation of response times in industrial WLANs, *IEEE Transactions on Industrial Informatics*, 3(3), 191–201, August 2007.
24. G. Cena, I. Cibrario Bertolotti, A. Valenzano, and C. Zunino, Industrial applications of IEEE 802.11e WLANs, in *Proceedings of the WFCS 2006*, *6th IEEE International Workshop on Factory Communication Systems*, Dresden, Germany, pp. 129–138, May 20–23, 2008.
25. G. Baumann, Controlling the wireless field, *The Industrial Wireless Book*, Issue 9:3. Available [online]: http://wireless.industrial-networking.com/articles/technical.asp.
26. OMRON, *WD30-ME/-SE/-ME01/-SE01 DeviceNet Wireless Units—Operation Manual*. Available [online]: http://www.omron.com/.
27. Institute of Electrical and Electronics Engineers, IEEE Standard for local and metropolitan area networks media access control (MAC) bridges, IEEE Std. 802.1D, June 2004.
28. L. Seno and S. Vitturi, Wireless extension of Ethernet powerlink networks based on the IEEE 802.11 wireless LAN, in *Proceedings of the WFCS 2006, 6th IEEE International Workshop on Factory Communication Systems*, Dresden, Germany, pp. 55–63, May 20–23, 2008.
29. Ethernet Powerlink Standardization Group, *Ethernet Powerlink Communication Profile Specification*, Version 2.0, 2003.
30. Defense Advanced Research Projects Agency, RFC 768, User Datagram Protocol, DARPA Std., August 1980.
31. G. Santandrea, A Profinet IO application implemented on wireless LAN, *IEEE WFCS 2006*, *Industry Day*, Torino, Italy, June 2006. Available [online]: http://wfcs2006.ieiit.cnr.it/.

32. 3Com, *3Com Wireless 7760 11a/b/g PoE Access Point User's Guide*, Prod. #3CRWE776075/WL-561. Available [online]: http://www.3com.com/.
33. Institute of Electrical and Electronics Engineers, IEEE 802-2 standard: Logical link control (with amendments 3, 6 and 7), IEEE Std., 1998.
34. The ZigBee Alliance, *ZigBee Specification*, December 2006. Available [online]: http://www.zigbee.org.

27

Wireless Sensor Networks for Automation

27.1 Introduction 27-1
WSN in Industrial Automation • Development Challenges • Reference Case • Development Aspects
27.2 WSN in Industrial Automation 27-2
"Classical" WSN • (Re)-Defining WSNs • Industrial Automation Applications
27.3 Development Challenges 27-6
Diverse Requirements • Choosing Communication Technology • Choosing Suppliers and Components • Pulling It All Together
27.4 Reference Case 27-13
Wireless Vibration Monitoring Application • Communication Standard • Communication Technology
27.5 Communication Standards 27-16
Standards Landscape • Technology Primer • Why Is ZigBee Not Enough?
27.6 Low-Power Design 27-25
Basic Principle • Sleep Modes • HMI • Energy Efficient Protocols
27.7 Packaging 27-29
IP Rating • Hazardous Environments (EX) • Other Environmental Considerations • Form Factor
27.8 Modularity 27-34
Levels of Modularity • Interfaces • Subsystem Behavior
27.9 Power Supply 27-38
Requirements • Power Management • Sources and Technology
References 27-43

Jan-Erik Frey
ABB Group

Tomas Lennvall
ABB Group

27.1 Introduction

The vision of ubiquitous or pervasive computing prescribes a paradigm shift, where the computing power is embedded in our environment rather than concentrated in desktop or laptop machines. This broad vision of the future has fuelled a number of narrowly defined research areas, among them wireless sensor networks (WSNs). WSNs originate from military applications such as battlefield surveillance, but are now being deployed in a large number of differing civilian application areas such as environment and habitat monitoring, healthcare applications, home automation, and traffic control. The intrinsic qualities of WSN such as self-powered, -organizing, and -healing make them highly

interesting also in the domain of industrial automation. However, industrial automation applications often have very distinct and stringent requirements, which necessitate that we loosen and extend the classical definition of a WSN for them to be more applicable in an industrial setting.

This chapter will provide an overview of typical industrial automation applications, explain how they differ from other domains, and highlight the key challenges when developing WSNs for the industrial automation industry. A reference case will be used throughout the chapter to exemplify these challenges and illustrate practical implications of design choices and requirements trade-offs. The chapter is organized in four main parts as outlined below.

27.1.1 WSN in Industrial Automation

We start off with a (re)-definition of WSN from an industrial automation point of view, and detail the key industry requirements and their impact on the core WSN technology.

27.1.2 Development Challenges

Following the description of applications and their requirements, we try to structure a generic development cycle or process that highlights key challenges of developing WSN devices. This section clearly shows how many different aspects that need to be considered, how intricate they are, and how intertwined with each other.

27.1.3 Reference Case

Here, we describe the application of the reference case including the environmental conditions, application requirements, and business motivation. The main elements of the design and implementation are described to provide a basis for comparison in the following sections on development aspects.

27.1.4 Development Aspects

The final sections in this chapter elaborates on the key development aspects described earlier such as standards, low-power design considerations, packaging, HW/SW modularity, power supply, and component choice.

27.2 WSN in Industrial Automation

27.2.1 "Classical" WSN

Wikipedia defines a WSN as "a wireless network consisting of spatially distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different locations" [1]. The devices, which are the focus of this chapter, are self-contained units comprising a microcontroller, power source (usually, but not always, a battery), radio transceiver, and sensor element (Figure 27.1).

WSN have some intrinsic qualities, which can be derived from the distributed autonomous nature of the devices.

27.2.1.1 Self-Powered

WSNs are self-contained units with an internal power source (usually, but not always, a battery). Because of the limitations of battery life, nodes are built with power conservation in mind, and generally spend large amounts of time in a low-power "sleep" mode.

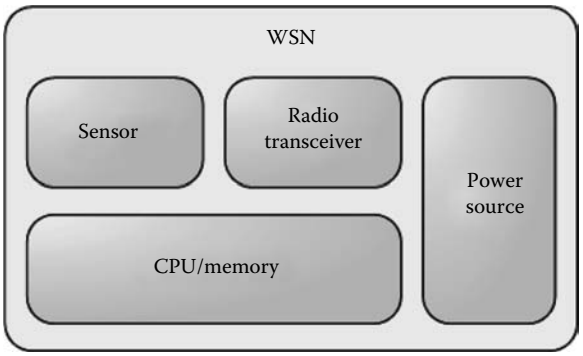


FIGURE 27.1 Components of a WSN device. (From Aakvaag, N. and Frey, J.-E., *ABB Rev.*, 2, 39, 2006. With permission.)

27.2.1.2 Self-Organizing

The nodes self-organize their networks in an ad hoc manner, rather than having a preprogrammed network topology. For example in wired network technologies, some designated nodes, usually with custom hardware (routers, switches, hubs, and firewalls), perform the task of forwarding data to other nodes in the network. In a WSN, typically each node is willing to forward data for other nodes, resulting in a dynamic network configuration based on the network connectivity.

27.2.1.3 Self-Healing

WSNs have the ability to self-heal, i.e., if one node goes down, the network will find new ways to route the data packets. This way, the network as a whole will survive even if individual nodes lose power or are destroyed. This type of dynamic reorganization can also be triggered by other conditions than node failure, such as maximum node power consumption and communication latency.

27.2.2 (Re)-Defining WSNs

Industrial applications differ from the definition above in a number of respects.

27.2.2.1 Cooperative Sensing Is Not Commonplace in Industrial Automation

First, and maybe most importantly, all sensors are crucial to the operation of the plant. This implies that losing one node is not an option, even if the overall network stays operational. A faulty node will have to be replaced. Increasing the availability by adding redundant sensors would be technically possible, but hardly feasible from a cost perspective. Although the cost of radio transceivers is continuously dropping, the majority of the cost of industrial-grade WSN devices is not driven by the cost of the transceiver alone. To this, you need to add the life cycle cost associated with installing and maintaining a device throughout its lifetime (which is a factor of at least 2–5 times longer than consumer products).

The notion of cooperative sensing prescribed by classical WSN is also not readily applicable to today's automation control systems, which are rather input and output (I/O) centric. Measurement points are not "dispersed" across the plant, but rather installed in very exact locations. The data are collected and fed back to the control system, where it is processed and analyzed. Adding a measurement point is associated with manual engineering and application programming before it can be utilized in the system.

The cooperative nature of WSN still adds value though, as setup and installation of the sensors becomes easier and the overall operation more reliable. Since the position of the measurement point might not be optimal from a radio frequency (RF) perspective, the self-organizing and -healing nature of WSN help alleviate communication disturbances related to radio propagation in industrial environments. In stead of investing lots of time in frequency planning and network design, performance of the network can be improved by adding router nodes that provide alternate/redundant paths for the wireless sensors.

27.2.2.2 Time Is of Essence

Time is money, also in the industrial automation world. Whereas a data packet in a standard WSN may spend an unknown time from its source to its destination, an industrial application will frequently require hard bounds on the maximum delay allowed. This can put some strict bounds on network topology, e.g., prescribing a star configuration rather than a multi-hop meshed network, and requires a careful trade-off between power conservation and response time. Although there is usually a strict bound on the maximum delay, the requirements can differ quite dramatically between applications. This makes a generic design of the WSN quite difficult and the "one size fits all" problem quite prevalent in industrial standards groups who seek to minimize the number of overlapping wireless communication standards.

27.2.2.3 Wireless in a Wired World

Although WSNs have made their entry on the industrial automation scene, the majority of sensors and devices have a wired infrastructure. This means that we need to cope with a mix of wireless and wired devices over a foreseeable future. Applications that are built upon a wired infrastructure can afford to "waste" bandwidth and make use of high performance networks to solve their mission critical goals. Simply adding a wireless sensor to the equation is often not straight forward.

A quite common mixed scenario on the other hand is to add wireless connectivity to already wired devices. This is done to provide access to diagnostic or asset data in the device in a nonintrusive manner, i.e., without modifying or interrupting the wired network.

Since wireless solutions in industry will tend to also have a wired infrastructure, the data will emanate from the sensors, and ripple through the network to some wired aggregation point. From here, it will in general, be transported over a high-speed bus to a controller. Apart from the classical mesh networking topology of WSN, we have two more common topologies in industrial settings (Figure 27.2).

In the star topology, the most prevalent topology today, the wireless nodes communicate with a gateway device that bridges the communication to a wired network. This network configuration will have greater chances of minimizing delays in the network and can thus be used in applications that have strict bounds on the communication delay. Another common intermediate solution of WSN is to have router devices (often mains powered) that communicate with the gateway. The sensors only need to perform point-to-point communication with the routers and can therefore remain simple and low power, while the range and redundancy of the network are improved.

27.2.3 Industrial Automation Applications

27.2.3.1 Industrial Plants, an Unfriendly Setting for WSN

The environmental conditions of industrial automation plants pose some heavy challenges on the development of embedded field devices in general, and specifically for WSN devices.

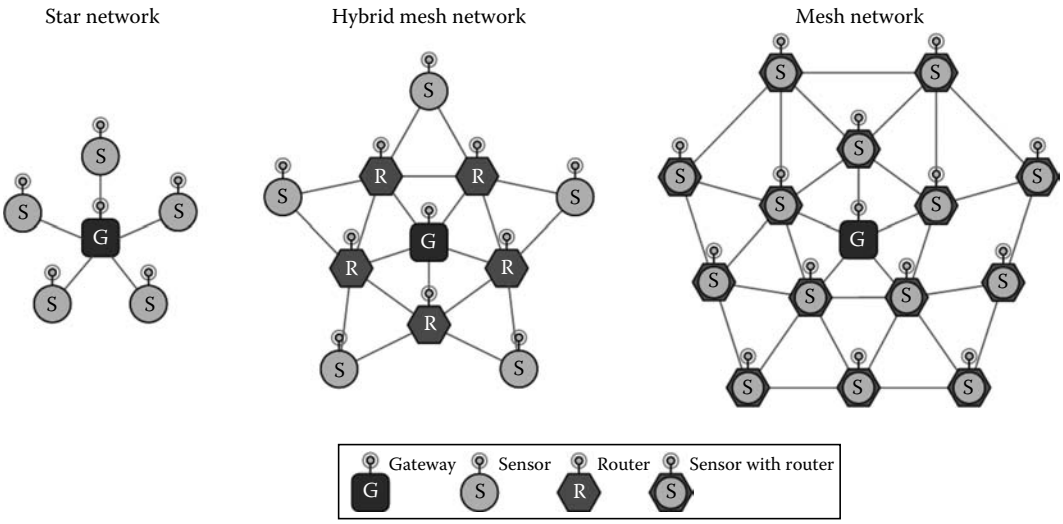


FIGURE 27.2 WSN topologies. (From Aakvaag, N. and Frey, J.-E., *ABB Rev.*, 2, 40, 2006. With permission.)

Industrial automation environments are characterized by

- **Harsh environments**: Extreme temperature, moisture, vibration, hazardous environments, steel constructions, and (moving) obstructions. Issues facing wireless communication in such environments are heavy multipath fading, fast/slow fading, coverage quality (due to reflection), and local variations in received power.
- **EMI**: Electromagnetic interference from electrical activity such as drives and welding, which cause noise in radio frequency bands.
- **Other users**: Disturbance by occupation of frequencies over time (WLAN, Bluetooth, ZigBee, etc.).

The following sections will elaborate more on the implications of these severe conditions on the embedded design of WSN devices.

27.2.3.2 Typical WSN Applications

The requirements of any WSN solution will always depend heavily on the particular application in mind. There are many promising applications for wireless technologies in industrial automation, ranging from monitoring of process parameters to mission-critical control applications. International Society of Automation (ISA [2]) defines six usage classes of wireless communication based on the criticality of the application, which directly governs the importance of the message response time (see Table 27.1).

WSNs in the traditional sense (i.e., low power, multi-hop, mesh networks) are today usually deployed in class 4–5 applications. This is logical since most WSN solutions are usually optimized for low-power consumption to increase battery's lifetime, and this stands in direct conflict with short and deterministic response times.

However, in real life, the boundaries between these classes are not always that clearly defined/identified. It is common that from a user's point of view, the application is a mixture of monitoring and control, making the requirements on message delay hard to specify. Defining a suitable wireless communication solution can thus be difficult unless we provide means to influence the network forming and optimization criterions.

TABLE 27.1 ISA-100 Usage Classes

Category	Class	Application	Description	
Safety	0	Emergency action	(Always critical)	
Control	1	Closed-loop regulatory control	(Often critical)	↑
	2	Closed-loop supervisory control	(Usually noncritical)	Importance of message timeliness increases
	3	Open-loop control	(Human in the loop)	
Monitoring	4	Alerting	Short-term operational consequence (e.g., event-based maintenance)	
	5	Logging and download-ing/uploading	No immediate operational consequence (e.g., history collection, sequence-of-events, preventive maintenance)	

27.2.3.3 Same-Same, but Different

Industrial automation is a broad industry and comprises many segments and application domains. A common way to divide the market is according to the type and characteristics of the process being controlled:

- Process automation (continuous processes)
- Discrete manufacturing (manufacturing of discrete units)
- Hybrid automation (mix of continuous and discrete)

The industry segments can be characterized by a number of process attributes that govern the requirements on wireless communication technologies, and consequently also the design and implementation. Table 27.2 below provides a description of key process attributes and their impact on the requirements and design of wireless technologies.

Although similar (bad) environmental conditions can be found throughout the various industry segments, there are distinct segment-specific requirements that have a dramatic impact on the wireless technologies. Figure 27.3 provides a comparison of the key process attributes within process automation and discrete manufacturing (note that the scale is logarithmic).

As illustrated in Figure 27.3, the two industry segments have quite different characteristics, and consequently also pose quite different requirements on the wireless technologies employed.

Even though industrial automation standardization activities in the wireless domain have started out with a quite holistic view of the processes and target applications, these distinct differences have led to the forming of separate working groups targeting process and discrete applications. For an overview of the industrial automation wireless standards landscape, see Section 27.5.

27.3 Development Challenges

The previous section presented a definition of the classical WSN, and contrasted that to the complex set of requirements and applications of industrial automation. These requirements and use-cases have a dramatic impact on the development of WSN devices.

Figure 27.4 depicts a generic development process.

Embedded development methodology and development processes in general are complex subject matters that merit a book of its own to do the topic justice. The process depicted above thus only aims at highlighting some key characteristics of WSN development projects:

1. A thorough requirements analysis is required in order to be able to select the right communications technology.
2. Choice of technology is not driven by the requirements alone. Factors such as standards compliance and legacy systems also have a large impact on the final choice.

TABLE 27.2 Impact of Process Attributes on Wireless Technologies

Process Attribute	Description	Impact on Wireless Technology
Device types	Predominant or common mix of devices, e.g., analog and/or discrete devices	The type of data (and consequently also amount) that is sent has a major impact on the design of the communication protocol. This in combination with the timing requirements sets boundaries on the frame/package size utilized.
Device count per unit	Typical number of devices in a manufacturing unit/production cell	The number of devices in a plant, their geographic distribution has a dramatic impact on the communication technology. The ability to cover a whole factory with wireless communication depends on a number of factors such as
Unit physical size (m)	The physical size of the a typical manufacturing unit/production cell	• Range of the wireless device/network, which is governed in turn by the reliable operation range, the disturbing range, and "frequency reuse" range
Units per plant	Number of manufacturing units/ production cells in a typical plant	• Number of wireless nodes per manufacturing unit • Number of coexisting wireless networks within reliable operation range without performance degradation
Device density (units per 1000 m^2)	Number of devices per 1000 m^2	
Devices per plant	Number of devices in a typical plant	
Production cycle length	Length of a typical production cycle	The production cycle length and startup times have a direct impact on the setup times for the wireless networks, such as network forming and adding/removing nodes.
Unit startup time	Startup time of a manufacturing unit/production cell	
Time: fast/subunit	Fast internal communication needs, e.g., drives loop	This attribute in combination with the node density has the greatest impact on the technology selection. Depending on the required update rate and equally important the latency (variation in communication delay), very radical differences in design of the wireless communication solution might be required.
Control loop	Time resolution of normal control loops	
Slow/diag./HMI	Time resolution for noncritical data such as diagnostics or HMI	
Field device cost	Typical cost of the field devices	The cost of the device to a large extent puts some upper barriers on the target cost of the wireless communication solution.
Installation cost/device cost	Cost of the installation in comparison to the cost of installing the device	This metric provides guidance on the target cost of installing and setting up the wireless infrastructure.
Automation technology	Type of control system utilized to automate the process	The use of a DCS vs. e.g., a PAC/PLC solution, to some extent governs the logical network structure. This is also reflected in the device count and unit count measures listed above.
Commonly used device networks	Typical wired infrastructure in today's plants	At some point, the wireless network has to interface with the existing network infrastructure. The employed wired network technology and topology have an impact on the way one could/should integrate the wireless system in the plant.
Power availability	Availability of line power/battery size/energy harvesting	The amount of energy that is available has a dramatic impact on the design of the wireless protocol (energy efficiency), as well as the possible duty cycle of the application.

3. Component selection to a large extent follows the choice of technology, but is many times already "given" by the platform choices of existing legacy systems.
4. Finally when designing, implementing, and testing the device, we need to pull all the bits together. This usually requires quite an iterative process with many trade-offs and redesigns due to a combination of inaccurate or incomplete requirements and/or component specifications.

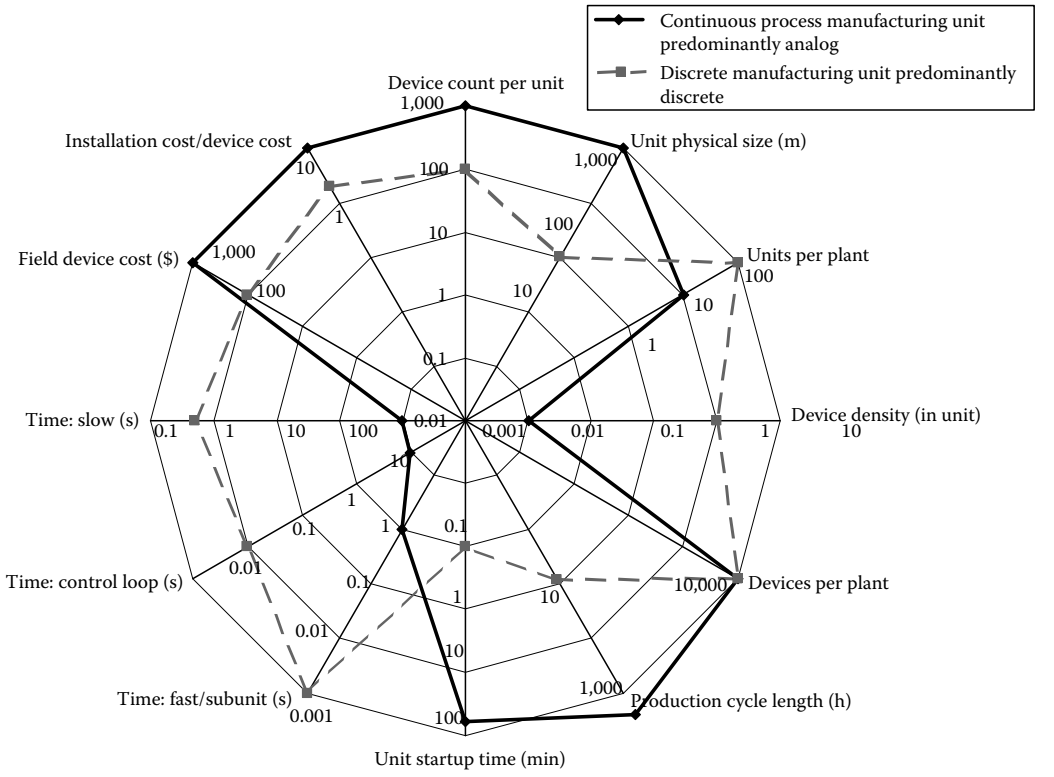


FIGURE 27.3 Process attributes.

27.3.1 Diverse Requirements

As highlighted in the previous section, the target industry and application can pose quite differing requirements on the choice of communication technology, as well as impose quite rigid restrictions on the embedded design. The requirements can be grouped into nonfunctional requirements that stem primarily from the target environment, and functional requirements given by the target application.

The environmental conditions of the target industry have a major impact on the boundary conditions of the embedded design and choice of communication technology. For example, factors such as heat, moisture, and vibration influence the packaging of the device, which in turn imposes additional requirements on the power consumption, as well as restrictions on the antenna design and power supply solution. If the device shall be used in hazardous environments, special consideration must be taken throughout the design process.

The application requirements more or less dictate the duty cycle of the WSN, which has a huge impact on choice of technology and a direct influence on the low-power design and required response time. In addition to the autonomous operation of the device, the human–machine interaction can pose additional requirements on external interfaces that influence both the communication solution as well as the packaging of the device.

27.3.2 Choosing Communication Technology

A clear view of the requirements helps eliminate obvious communication solutions. However, it is common that one needs to make a trade-off in the end to be able to address the main requirements that are driving the application.

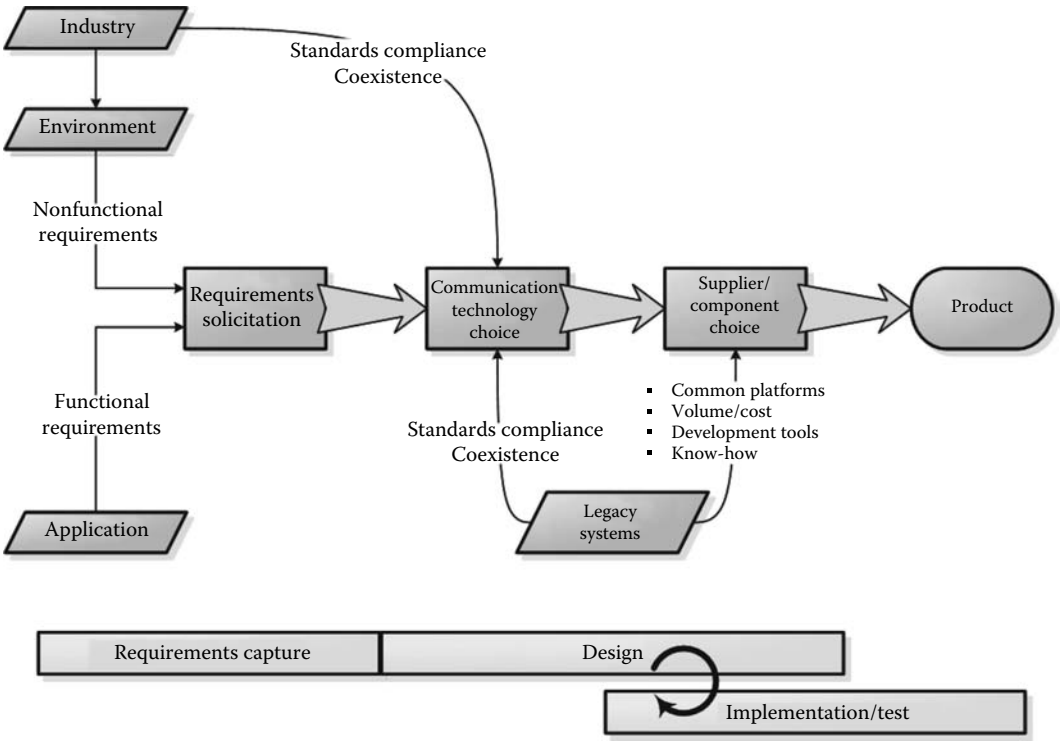


FIGURE 27.4 Principal development process.

Many industries, customers, and suppliers have adopted their own communication standards, or when possible refer to international standards. If a different technology shall be acceptable, as a minimum requirement, it must be able to coexist with other technologies without unduly compromising the performance of installed devices/systems.

27.3.3 Choosing Suppliers and Components

The choice of technology has a direct impact on the availability of suitable SW/HW components, and thus also on the total development cost/time. Since industrial devices have a quite long lifetime, legacy systems have a major impact on choice of technology as well as components. Many suppliers have streamlined their development processes and base their product portfolios on standardized platforms. This has great benefits in terms of volume/cost of components and development tools, as well as spreading of best practices and know-how. Introducing a new SW/HW component or platform is often a demanding and time-consuming task, and always represents an additional risk. You can never be quite sure that the new component will behave in the same manner as those components that have undergone years of development/refinement and are proven in the field. It might therefore be better to utilize a well-known component, and suffer/deal with the performance losses of utilizing a "nonoptimal" component from a WSN perspective.

27.3.4 Pulling It All Together

The development process depicted in Figure 27.4 gives the impression that the development follows a rather linear waterfall-type model. Even if you might start out that way, due to the complexity of the requirements and the boundary conditions imposed by industry and legacy systems, the design/implementation/test phase usually reverts to a highly iterative process. Developing WSN

Impact on	Communication standards	Low-power design	Packaging	Modularity	Power supply	Component selection
Communication standards	✓ Coexistence	✓ Upper bounds ✓ Lower bounds	✓ HW/SW Interfaces	✓ Functional integration		✓ Com stacks ✓ Supported HW architectures
Low-power design			✓ HMI options	✓ Functional integration	✓ Peak power ✓ Average power	✓ Functionality ✓ On/off timing ✓ Second source
Packaging		✓ HMI power consumption ✓ IS adaptations		✓ Integration ✓ Safety barriers ✓ Interfaces	✓ Peak power ✓ Heat build-up ✓ Sparks	✓ Battery ✓ Antenna design ✓ Surface temp.
Modularity		✓ Functional integration ✓ Interfaces	✓ Form factor ✓ Interfaces		✓ Make vs. buy	✓ Interfaces ✓ Footprint ✓ Debugging
Power supply		✓ Duty cycle	✓ Form factor ✓ Battery change	✓ Make vs. buy		✓ Battery technology

FIGURE 27.5 Development aspect matrix.

devices for industrial automation applications often comprises a set of "chicken and the egg" problems, where the order of design choices can result in very different outcomes in the final design. A key success factor is thus to have a very clear view of the driving requirements, i.e., the absolute "must haves," stick to them, and let the other lower priority requirements, the "should haves," undergo the inevitable trade-off exercise.

Figure 27.5 illustrates the interdependencies of key development aspects of WSN devices, and shows how one design aspect (a row) affects other design aspects (the columns).

The following sections provide a walk-through of the table to demonstrate how design choices in one dimension can, directly or indirectly, affect other dimensions of the design (and vice versa).

27.3.4.1 Communication Standards

Most customers strive toward the use of standards when building their installations. This secures interoperability and availability of suppliers and leads to higher volumes (and ultimately lower cost). The use of a standard is thus often mandatory to be able to compete on the market. One of the key requirements of wireless devices in industry is coexistence. Hence, this needs to be considered when choosing a specific standard.

Once selected, the standard in itself will impose upper/lower bounds for the device in terms of power consumption. Regardless whether you employ low-power electronics and a clever sleep–wake-up scheme, the communication protocol has a vital impact on the final power consumption of the system. Some communication protocols are notoriously inefficient and no smart embedded programming in the world will help get the consumption down to an acceptable level.

The standard can also impose certain physical interfaces, which will have a direct impact on the design of HW/SW as well as the packaging of the device. For example, the wireless highway addressable remote transducer (WirelessHART) standard [3] requires that you provide a service port on your devices.

The scope and architecture given in the standard indirectly also prescribes the level of integration of HW/SW. This is of course highly linked to the available commercial offering that supports the standard, such as radio transceivers and communication stacks.

So, "simply" by choosing a specific standard, many design choices have been made that will drive the design of the device. In Section 27.5, we will dig deeper into the wireless standards jungle, and provide some insight to the basic mechanisms of modern WSN solutions targeted toward industrial automation.

27.3.4.2 Low-Power Design

One important aspect of WSN is to keep the power consumption of a node at a minimum, while at the same time providing the best possible performance to the system users. The low-power nature of WSN poses serious restrictions on the power consumption of the HW as well as on the SW architecture.

Integrating functionality can be one way to lower the overall power consumption, but can put severe limitations on the modularity of the system. This not only affects the HW/SW maintenance of the device, but may also limit your options concerning what to power up/down and when, which in turn can affect the overall power consumption as well as performance of the device negatively.

Although WSN devices are autonomous entities, a local human–machine interface (HMI) is often useful or necessary during commissioning and troubleshooting. The specific low-power design directly limits your options concerning the use of a local HMI, e.g., liquid crystal display (LCD), light emitting diode (LED), buttons, etc.

Designing for low power consumption involves choosing low-power components. This may seem trivial, but is often a complex issue as it usually involves compromises on the performance. Typically, a low-power CPU runs on reduced clock cycle with fewer on-chip features than its more power hungry counterparts. The trick is to choose elements with just enough performance to do the job. In addition, "over specifying" components to create margins in your design can lead to cost and maintenance issues as the number of suppliers tend to shrink rapidly.

Finally, the low-power design more or less dictates the required average and peak power that needs to be available in the system and this may have some serious consequences on the design of the power supply solution, including choice of power source.

Less really intrinsically safe (IS) more in the WSN world, and the more you take away, the more challenging the design task becomes. In Section 27.6, we invest some more "energy" in describing the intricacies of low-power design.

27.3.4.3 Packaging

The packaging of the WSN device represents one of the most costly parts of the total product. The environmental conditions and the device rating (IP class, EX, temperature range, etc.) all influence the required ruggedness of the device and prescribe what material to use in the housing and how well it must be sealed off to protect the internal components from dust and moisture.

The use of a local HMI, whether it is an LCD or a single LED, can have a dramatic impact on the overall power consumption of the device and thus needs to be accounted for in the low-power design.

If the device is to be able to operate in a hazardous environment, great care needs to be taken throughout the design of the device. An intrinsically safe (IS) design is commonly required. The basic principles of an IS design are simple and easy to grasp, but challenging to achieve in practice. It requires long experience in the field as well certification by a notified body. Making IS adaptations can have a dramatic impact on the power consumption and the form factor of the device (due to additional protection and power limiting components). Often the maximum power that can be provided by the power supply needs to be limited. The major requirement for batteries operating in ATEX applications is for intrinsic safety. Specifically, this calls for a battery design that prevent both excessive heat buildup and the danger of an electric spark. The modularity of the device is governed by the need to define barriers with clear interfaces in the design to contain and transfer energy in a controlled and safe manner. The maximum surface temperature of any component on a circuit board under fault conditions must be considered.

The form factor of the device components is often given indirectly, either through product design guidelines to foster harmonization and common look and feel or by the requirement to utilize an already existing housing. This means that you often have a given space where you need to squeeze in your HW components, which can have a dramatic impact on the required level of component integration (single chip solutions, double-sided printed circuit boards (PCBs), etc.). It is common

that the size of one specific component can force a complete PCB redesign to fit within the given dimensions. Typical components that drive the PCB design are capacitors and batteries.

Apart from component selection and integration, the form factor also has a direct influence on the antenna design, which is very sensitive to the orientation and proximity of other electronic components.

Appearance is everything! This actually holds true to a large extent when it comes to embedded design of industrial WSN devices. In Section 27.7, we find out that what counts is the "inside," but the appearance of the device (size, shape, material, and weight) does have a major impact on the final design.

27.3.4.4 Modularity

The lifetime of the devices is a key parameter to consider. The office/consumer industry is the main driver for wireless technologies today, with high volume applications. Industrial automation devices, however, tend to have a much longer lifetime than consumer products. This means that special care needs to be taken when integrating wireless components into industrial devices. A modular (hardware and software) design is crucial in enabling effective maintenance of devices—which are built on standard commercial off-the-shelf (COTS) components—throughout their lifetime.

Modular design is necessary in order to reuse elements, but it might seriously limit your control over individual HW/SW functions and thus influencing the overall sleep–wake-up scheme of your device. Proper interfaces are thus vital to mediate between reuse and performance, something that can be hard to influence when COTS components are used.

The form factor of commercial components or subassemblies naturally has a direct impact on the packaging of the device, but again, the interfaces are just as important, and need to be sufficiently general to allow portability.

One classical example of the separation of modules is the split between the communication protocol and the application software. The latter is invariably written by the device vendor, but the former is frequently purchased from a third party. Embedding these two components onto the same microcontroller can be difficult. We also run the risk of suboptimizing, i.e., the two software modules are optimized (with respect to power, performance, code size, etc.) individually. This does not necessarily give a globally optimum solution. Even more complex is handling new releases, bug fixes, and documentation when the software running on the same processor has several sources.

In addition to reuse, modularity is also a means to minimize the design risks. For example, to avoid electro magnetic compatibility (EMC) problems in your power supply design, it is often beneficial to buy predesigned and tested COTS components such as a DC–DC converter. If the device is going to be produced in large volumes, the cost of these COTS components can be prohibitive though, forcing you to make your own low-level design. Component specification can be another limiting factor when utilizing available designs, e.g., if you require a large operating range.

The sum of the parts can equal more than the individual parts together, at least from a product maintenance point of view. However, great care must be taken to ensure that sum of the parts also provides adequate performance. In Section 27.8, we cut up the WSN device into logical pieces and look how we can put them back together again without unduly compromising the control over the final behavior of the device (power consumption, latency, etc.).

27.3.4.5 Power Supply

The power supply is a central part of the WSN device; it is the "motor" that provides power to the various parts of the system. The challenge is the same as the development team has regarding the final product, to provide the right level, with the right quality, at the right time.

The design of the power supply can put direct bounds on the duty cycle of the application. For example, in an energy-scavenging solution, the harvested energy is usually stored in an intermediate

storage. This enables the power supply to provide an even power supply level in spite of a varying "incoming" power, and to handle intermittent peak consumptions. The size of this intermediate storage and the time it takes to charge it directly impact the "maximum" duty cycle.

Although battery technology is progressing, the battery still constitutes a major part of the footprint of the device if we are to reach the 5–10 years lifetime that are typically quoted. The same goes for the energy-scavenging solutions, which usually contain some form of intermediate storage as well.

Typical parameters to consider when choosing battery technology are operating temperature range, energy density, nominal voltage, and Watt hours (for a given cell size). However, there is more than meets the eye here as well. For example, in some types of lithium technologies, passivation may lead to voltage delay, which is the time lag that occurs between the application of a load on the cell and the voltage response. This behavior becomes more pronounced when you have very low duty cycles (since the passivation layer grows thicker).

If battery change in the field is to be supported, special care needs to be taken to allow separation of the battery from the device. This is especially tricky if the device is going to be deployed in a hazardous environment, where any spark must be prevented.

Low-power design is all about powering up/down various parts of the system. In Section 27.9, we find out that there is a lot more to consider than on/off.

27.4 Reference Case

As with other resource-constraint embedded systems, "the devil is in the details." In this section, we present one specific use-case of WSN in industrial automation. This application will be used as a reference case for the remainder of the chapter to exemplify some of the details that really matter when making the design choices.

The application is "wireless condition monitoring of AC motors using vibration analysis." In the following sections, all references to this specific use-case will be highlighted with a border:

Wireless Vibration Monitoring Case Example

27.4.1 Wireless Vibration Monitoring Application

The goal of the vibration monitoring application is to detect early signs of bearing degradation on small AC motors (less than 400 kW), located on offshore oil/gas rigs, in order to be able to schedule maintenance. The motors drive equipment such as pumps, air compressors and fans, and the load conditions as well as the operating environmental conditions will vary (Figure 27.6).

The application consists of several parts as illustrated in Figure 27.7: vibration-monitoring sensor nodes, repeater nodes, a gateway, and an asset management system (vibration analysis software).

For the remainder of the chapter, we will only refer to the WSN part of the application, which consists of the sensor and router nodes. The gateway is actually also part of this but is considered to be part of the infrastructure and often regarded as a COTS component. It is (usually) mains powered, and does not face the same design challenges as the resource-constraint sensor nodes. The need for repeater nodes depends on the communication technology and the operating environment of the WSN.

The WSN part of the application has the following design goals and requirements:

- **Low cost**: the sensor/repeater nodes are very restricted in what they are allowed to cost, which has a large impact on component selection and packaging. This also has the effect that the number of nodes that will be deployed in the field should be minimized; i.e., the range of the wireless communication should be as long as possible.

FIGURE 27.6 Wireless vibration sensor.

- **EX certification**: the environment on oil and gas rigs is hazardous. The sensor and repeater nodes must be certified to operate in an EX Zone 1 area, i.e., nodes will intermittently be subjected to flammable material (oil or gas), and must therefore limit the energy of any sparks and surface temperature.
- **Nonintrusive (easy to mount and retrofit)**: the sensor/repeater nodes must be easily mounted on the motors. Some motors have mounting locations prepared while others do not, and the mounting constructions also differ from case to case. The orientation of the sensor element and location on the motor (preferably close to the bearing) matters from a vibration analysis point of view, which can be in direct conflict with an optimal antenna positioning. All of this must be taken into consideration for the physical design of the sensor nodes.
- **Battery powered and a long battery life**: the requirement is that a sensor node should have atleast a 5 year lifetime without any battery change.
- **Duty cycle**: each sensor node should perform one vibration measurement per "two weeks" and send it to the asset management system for analysis. This is a bit opposed to the "typical" view of a WSN, which sends small amounts of data "fairly often" (minutes, hours, or days).
- **Operating environment**: the temperature can vary from well below zero up to, or even over, 100 °C. In addition, the wireless communication solution must be able to cope with the possible interfere caused by the many metal constructions found on an oil rig.
- **Security**: authenticity (making sure data comes from the correct sensor node) and integrity (making sure no one has tampered with the data) is very important. Therefore, it is also important that we only allow authorized nodes to join the network.

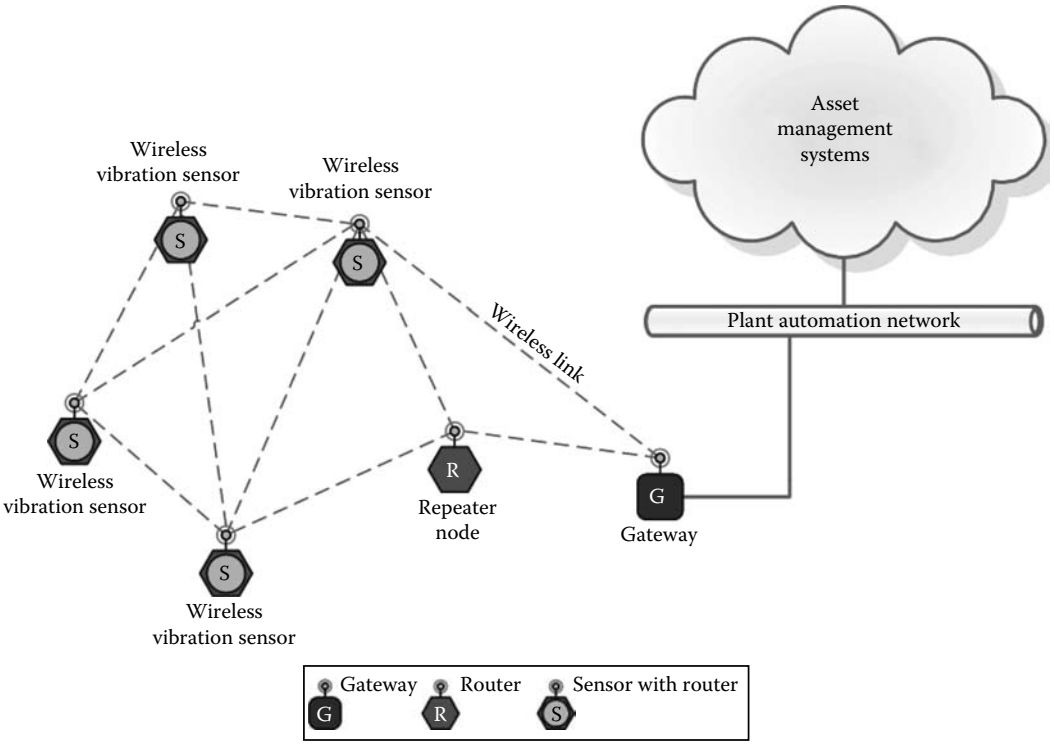


FIGURE 27.7 Wireless vibration monitoring system.

Latency is not a critical requirement for the application because the data are not part of a process control loop. Reliability on the other hand is a much more important requirement, because missing data will result in a flawed view of the actual motor condition.

27.4.2 Communication Standard

WirelessHART was the selected communication technology for the vibration monitoring application. It is currently the only industrial wireless communication standard (released September 2007) that is available, and it is explicitly designed with industrial applications and requirements in mind.

ZigBee is also a currently available wireless communication standard, but was not considered sufficient for operation in industrial environments, especially an environment such as the one on an offshore rig. Basically, it lacks effective means with which to alleviate the effects of interference and fading. We provide more details on ZigBee's shortcomings in industrial environments in Section 27.5.3.

27.4.3 Communication Technology

As mentioned, WirelessHART is the chosen communication standard for the vibration monitoring application, because it is currently the only available wireless communication standard designed for industrial applications.

However, because it was just recently released, technology suppliers have not yet provided any components supporting the standard. This has led to a decision to use a proprietary technology, as an intermediary solution, until WirelessHART components become available. This leads to another

FIGURE 27.8 DUST Networks SmartMesh MoC solution. (From DUST Networks, Inc., Media Kit—Product Images. http://www.dustnetworks.com, 2008. With permission.)

important decision we must make; we must select a proprietary solution that will provide/enable an easy migration to WirelessHART.

In the end, the decision fell on a solution offered by DUST Inc. Networks [4] called SmartMesh-XD. It is a System-on-Chip (SoC) WSN solution called Mote-on-Chip (MoC), see Figure 27.8. The SmartMesh-XD MoC (DN2140/2040) is a "black box" solution, which hides all details related to the wireless communication from the rest of the application.

DUST Networks will also provide a WirelessHART-based solution (SmartMesh IA-500), which uses identical hardware to the SmartMesh-XD MoC (including the interface), thus only requiring a firmware upgrade in order to become WirelessHART compliant. This will allow for a smooth transition to WirelessHART.

SmartMesh-XD is IEEE 802.15.4 standards compliant and operates in the 900 MHz and 2.4 GHz ISM bands. It is a robust WSN technology that combines low-power radio technology with the Time Synchronized Mesh Protocol (TSMP).* TSMP, which was also the basis for the WirelessHART standard, provides features such as time division multiple access (TDMA), frequency hopping, and redundant paths (mesh), which make it suitable for the harsh operating environment of the application. The SmartMesh-XD communication uses a concept called a network frame. The network frame is a fixed length, cyclically repeating, sequence of time slots (TDMA), in which all nodes have been assigned dedicated slots for message transmission and reception.

In the following sections, we highlight the impact of the decision to use SmartMesh-XD on the design of the application, as well as implications for future migration to WirelessHART.

27.5 Communication Standards

The advent of WSNs brings many new and exciting technologies into the world of industrial automation. There are currently a number of initiatives underway for standardizing WSN for industrial use. General trend on the market is consolidation of suppliers and migration toward standards, most noticeably the release of WirelessHART and the development of ISA100.11a. Additionally, a major automation network organization, PNO, has decided to use WirelessHART as a base for its wireless process automation solution.

* TSMP is a proprietary WSN protocol developed by DUST Networks.

TABLE 27.3 Comparison of Key WSN Standards in Industrial Automation

	ZigBee	WirelessHART	ISA100.11a
Availability	June 13, 2005	September 7, 2007	Draft
PHY layer	IEEE 802.15.4	IEEE 802.15.4	IEEE 802.15.4
Standard specifies	Layers 3 and 7	Layers 2–4, and 7	Layers 2–4, and 7
Frequency	868 MHz 915 MHz 2.4 GHz	2.4 GHz	2.4 GHz
RF transmission	DSSS 2.4 GHz: 16 channels 868/915: 1 + 10 channels	FHSS 2.4 GHz: 16 channels	FHSS 2.4 GHz: 16 channels
Media access	CSMA/CA optional GTS	TDMA, optional CSMA/CA shared slots within TDMA slot	TDMA
Network topology	Mesh, star, hybrid	Mesh, star, (hybrid possible, but not specified)	Mesh, star, hybrid
Bit rate	250 kbps	250 kbps	250 kbps
Security	Optional: symmetric keys, AES-128 encryption	Symmetric keys, AES-128 encryption	Symmetric keys, AES-128 encryption. Optional use of asymmetric keys
Coexistence	CSMA/CA	Optional channel black listing, optional CCA, settable transmission power levels	Optional channel black channel assessment (CCA), settable transmission power levels

Source: Lennvall, T., Svensson, S., and Hekland, F., in *7th IEEE International Workshop on Factory Communication Systems (WFCS 2008)*, May 21, 2008, Dresden, Germany. With permission.

27.5.1 Standards Landscape

Table 27.3 provides an overview of the three main WSN standards targeting the industrial (process) automation market. Most fieldbus organizations refer to either of these standards when they think about how to integrate WSN into their wired bus infrastructure. In addition, there are proprietary solutions out on the market, which could be considered de facto standards in certain niche applications.

Below follows some more details on the background of these standards. For a more technical description of the underlying technologies that are employed, see Section 27.5.2.

27.5.1.1 ZigBee

One of the best-known standards is the ZigBee standard; a low-power, -cost, and -data rate wireless specification, which targets home appliances, toys, industrial applications, building automation, and the like. The ZigBee standard is developed, managed, and promoted by the ZigBee Alliance [5]. The scope or focus of the ZigBee Alliance is on defining the network, security, and application software layers, providing interoperability and conformance testing specifications, and promoting the ZigBee brand.

27.5.1.2 WirelessHART

WirelessHART is the only industrial WSN standard currently available. It was released during September 2007 [3], and is a secure and reliable wireless technology for industrial applications ranging from class 2 to 5 (see Table 27.1), i.e., from control to process measurement/monitoring. WirelessHART extends the well-known HART standard into the wireless domain, enabling new market areas for the large number of HART users. HART communication [3] is a bidirectional industrial field communication protocol used to communicate between intelligent field instruments and host systems. HART is the global standard for smart process instrumentation and the majority, more than 25 million of smart field devices installed in plants worldwide, are HART-enabled.

27.5.1.3 ISA100

Another ongoing initiative is ISA100* (formerly called ISA-SP100) [2]. Originally, the intent of ISA100 was to not standardize all elements in the system, instead only to specify the upper levels in the stack (i.e., the open standards interface [OSI] stack), with a number of potential lower level implementations. This scope has been changed to include all levels of the stack as part of the specification.

The ISA100 workgroup also contained two subgroups that investigated wireless solutions for different use cases: SP100.11a aimed for control use cases, while SP100.14 targeted monitoring use cases.

Compared to WirelessHART, ISA100 takes a more generic path as an industrial WSN technology. It will support the transportation of many different fieldbus formats (where HART is one of them). ISA100 aims to support industrial applications in the range from class 1 to 5, i.e., from critical control to monitoring.

The standard is currently (February 2008) at a preliminary draft stage, and is expected to be released by the end of 2008.

27.5.1.4 PNO WSAN

The PNO (Profibus Nutzerorganization, [7]) formed a work group, wireless sensors and actuator networks (WSAN), with the purpose of investigating suitable wireless communication technologies for their two areas of interest: process automation and factory automation.

A decision to use WirelessHART as a base for process automation was taken by the end of 2007, but a suitable technology has not yet been selected for factory automation.

27.5.1.5 6loWPAN (IETF RFC 4944)

6loWPAN is an acronym for "IPv6 over low-power wireless personal area networks," which is a protocol for transmission of IPv6 packets over IEEE 802.15.4 networks. This technology will enable the extension of the typical WSN structure into an IP-based system, basically allowing any computer in the automation plant to access any WSN device. It also allows reuse of existing IP network tools developed for home and office use, such as commissioning and network debugging support, etc.

27.5.1.6 Proprietary Solutions

There are many suppliers of proprietary WSN solutions, who offer a wide range of solutions, from bare technology/components (stack and hardware) to complete WSN systems. There are also industrial automation suppliers who offer WSN solutions as part of their automation system.

Below is just a small selection of all the different products and solutions available on the market today:

- ArchRock [16] provides a complete 6loWPAN solution, IP-based sensor networking, which communicates using IEEE 802.15.4 (i.e., it operates in the ISM bands).
- Omnex Controls [17] provides devices, which uses a proprietary frequency hopping spread spectrum (FHSS) technology that operates in either the 900 MHz or 2.4 GHz ISM band.
- DUST Network [4] (see Section 27.4.3 for more details) is an FHSS solution, which is also based on IEEE 802.15.4.

* International Society of Automation (ISA).

- Accutech [18] also supplies an FHSS, 900 MHz ISM band solution for its wide range of devices.
- ABB Ltd. pioneered the use of wireless communication for discrete manufacturing control applications with their wireless interface to sensors and actuators (WISA) technology [19]. WISA provides real-time wireless communication based on a TDMA protocol, combined with frequency hopping based on the physical layer of IEEE 802.15.1.

27.5.2 Technology Primer

27.5.2.1 ZigBee

27.5.2.1.1 Architecture

ZigBee is a specification for a cost-effective, low-rate, and low-power wireless communication protocol for home automation, monitoring, and control. It aims to provide short-range wireless networking, which is scalable, self-organizing, and secure, while providing battery life up to 2 years. Although having existed since late 2004, ZigBee is yet to prove its success, at least in the industrial domain where reliability and security are of utmost importance.

ZigBee is a specification for the higher protocol layers, and builds upon the physical (PHY) and medium-access control (MAC) layers in the IEEE 802.15.4 specification [6], as seen in Figure 27.9.

27.5.2.1.2 Basic Functionality

Mesh networking topology is supported and routing is achieved through the ad hoc on-demand distance vector (AODV) algorithm. This means that the devices themselves that are responsible for route discovery, and that peer-to-peer communication is possible. In a ZigBee network, all nodes

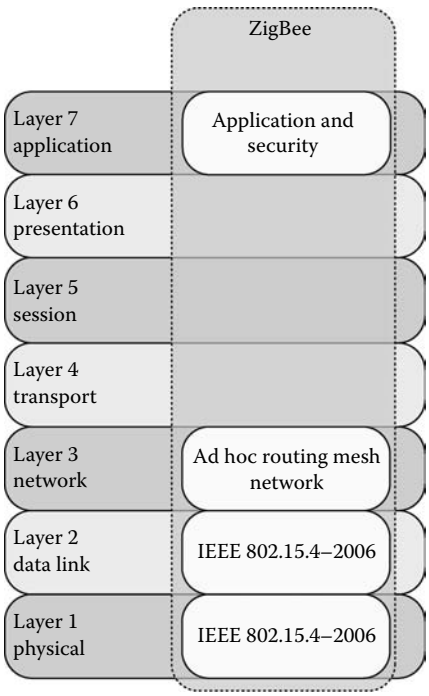


FIGURE 27.9 ZigBee protocol stack.

share the same channel, and frequency agility is minimal. There is no frequency hopping, and the only option is to scan for a channel with the least amount of interference at startup.

There are two classes of network devices in ZigBee: full-function devices (FFD) and reduced-function devices (RFD). The former can route messages in mesh networks and act as the network coordinator, whereas the latter can only communicate with one FFD in a star network setup.

ZigBee can operate in both beaconed and nonbeaconed mode. In the beaconed mode, the nodes are to some extent synchronized and the superframe is divided into 16 slots. The slots in the frame are generally contention-based, using carrier-sense multiple access with collision avoidance (CSMA/CA). There is an option to dedicate up to seven of these slots to specific nodes in order to increase determinism, so-called guaranteed time slot (GTS). However, support for this is not mandatory and use of this feature might break interoperability.

27.5.2.1.3 Security

In the 2006 version of the specification, security is not mandatory. However, support for authentication, integrity, and encryption for both network and application layer is present. MAC layer security available through 802.15.4 is not explicitly addressed in the ZigBee standard, and its use might break interoperability between different vendors' products. Replay attacks are protected against using sequential numbering. ZigBee makes use of the security mechanisms in 802.15.4; Counter with CBC–MAC* (CCM) with AES-128 encryption, but with the option to employ encryption-only or integrity-only.

Three key types are used: master key, link key, and network key. The master key is necessary to be able to join the network. The link key is used for end-to-end encryption and would by that provide the highest level of security at the price of higher storage requirements. The network key is shared between all devices, and thus presenting a lower level of security, though with the benefit of reduced storage requirements in the devices. All keys can be set at the factory, or be handed out from the trust center (residing in the network coordinator), either over the air or through a physical interface. For commercial grade application, the trust center can control the joining of new devices and periodically refresh the network key.

27.5.2.2 WirelessHART

27.5.2.2.1 Architecture

WirelessHART was designed based on a set of fundamental requirements: it must be simple (e.g., easy to use and deploy), self-organizing and -healing, flexible (e.g., support different applications), scalable (i.e., fit both small and large plants), reliable, secure, and support existing HART technology (e.g., HART commands, configuration tools, etc.).

Figure 27.10 shows that the architecture of WirelessHART mimics the OSI layer design. WirelessHART is based on the PHY layer specified in the IEEE 802.15.4-2006 standard [6],† but specifies a new data-link (including MAC), network, transport, and application layers. The figure also shows the backward compatibility of WirelessHART because it shares the Transport and Application layer with HART.

27.5.2.2.2 Basic Functionality

WirelessHART is a TDMA-based network. All devices are time synchronized and communicate in prescheduled fixed length time-slots. TDMA minimizes collisions and reduces the power consumption of the devices.

---

* Cipher Block Chaining-Message Authentication Code.

† WirelessHART only supports the globally available 2.4 GHz ISM frequency band.

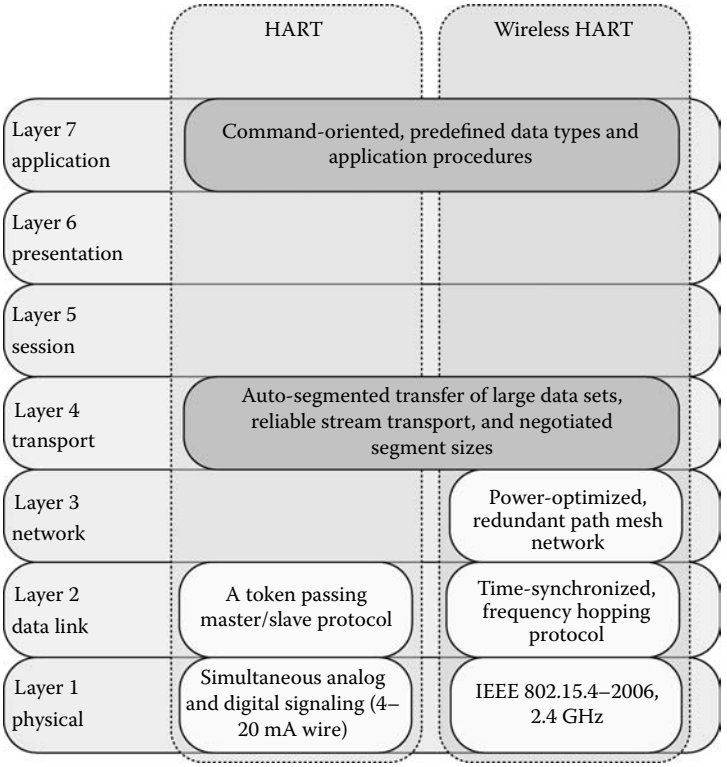


FIGURE 27.10 HART and WirelessHART protocol stacks.

WirelessHART uses several mechanisms in order to successfully coexist in the shared 2.4 GHz ISM band:

- *FHSS* allows WirelessHART to hop across the 15 channels defined in the IEEE 802.15.4 standard in order to avoid interference.
- *Clear Channel Assessment* (CCA) is an optional feature that can be performed before transmitting a message. CCA works like CSMA/CA but without using an exponential back-off. This improves coexistence with other neighboring wireless systems.
- *Transmit power level* is configurable on a node level.
- Mechanism to disallow the use of certain channels, called "Blacklisting," is available.

All of these features also ensure WirelessHART does not interfere with other coexisting wireless systems that have real-time constraints.

All WirelessHART devices must have routing capability, i.e., there are no reduced function devices like in ZigBee. Since all devices can be treated equally, in terms of networking capability, installation, formation, and expansion of a WirelessHART network become simple (self-organizing).

WirelessHART forms mesh topology networks (star networks are also possible), providing redundant paths, which allow messages to be routed around physical obstacles, broken links, and interference (self-healing). Two different mechanisms are provided for message routing: "Graph routing" uses predetermined paths to route a message from a source to a destination device. To utilize the path redundancy, Graph routes consist of several different paths between the source and destination devices. Graph routing is the preferred way of routing messages both up- and downstream in a WirelessHART network. "Source routing" uses ad-hoc-created routes for the messages without providing

any path redundancy. Source routing is therefore only intended to be used for network diagnostics, not any process-related messages.

27.5.2.2.3 Security

Security is mandatory in WirelessHART; there is no option to turn it off or to scale it up/down. WirelessHART provides end-to-end and hop-to-hop security measures through payload encryption and message authentication on the network and data-link layers. However, the security measures are transparent to the application layer.

WirelessHART uses CCM* mode in conjunction with AES-128 block cipher, using symmetric keys, for the message authentication and encryption.

A set of different security keys are used to ensure secure communication. A new device is provisioned with a "Join key" before it attempts to join the wireless network. The Join key is used to authenticate the device for a specific WirelessHART network. Once the device has successfully joined the network, the network manager will provide it with proper Session and network keys for further communication.

The actual key generation and management are handled by a "plant wide" "Security manager," which is not specified by WirelessHART, but the keys are distributed to the Network devices by the Network manager. A "Session key" is used on the Network layer to authenticate the end-to-end communication between two devices (e.g., a Field device and the Gateway). Different Session keys are used for each pair-wise communication (e.g., Field device to Gateway, Field device to Network manager, etc.). The data-link layer (DLL) uses a *network key* to authenticate messages on a one-hop basis (e.g., to and from neighbors). A well-known network key is used when a device attempts to join the network, i.e., before it has received the proper network key. Keys are rotated based on the security procedures of the process automation plant.

27.5.2.3 ISA100

The following description of ISA100.11a is a snapshot of what the standard looks like at the time of writing (March 2008). The standard will very likely undergo some (minor) changes before the final version is released.

27.5.2.3.1 Architecture

ISA100.11a is designed to fit the needs of many industrial protocols, such as HART, Profibus, Foundation Fieldbus, etc. Its scope also includes parts of the plant backbone network.

Figure 27.11 shows an ISA100.11a network that includes a part of the backbone. The leftmost part, from the backbone routers and to the left, is called the "data-link" layer subnet, i.e., it consists of non-routing devices, routing devices, and backbone routers. However, in the simplest network scenario, the backbone routers are replaced by one gateway and manager device, which is directly connected to the plant automation network.

Figure 27.12 shows the architecture of ISA100.11a described in terms of the OSI reference. The "physical" layer is based on IEEE 802.15.4-2006 and so is the MAC part of the (DLL). The upper part of the DLL layer handles all communications aspect in the DLL subnet part of an ISA100.11a network. A DLL subnet corresponds to a WirelessHART network, i.e., a time synchronized, frequency hopping mesh network composed of field devices. The "network" layer is based on a subset of the 6loWPAN (IETF RFC 4944) specification, i.e., sending IPv6 packets over 802.15.4 networks. The transport layer is also based on 6loWPAN, which specifies the use of UDP for end-to-end delivery of packets. Note

* Counter with CBC–MAC (corrected).

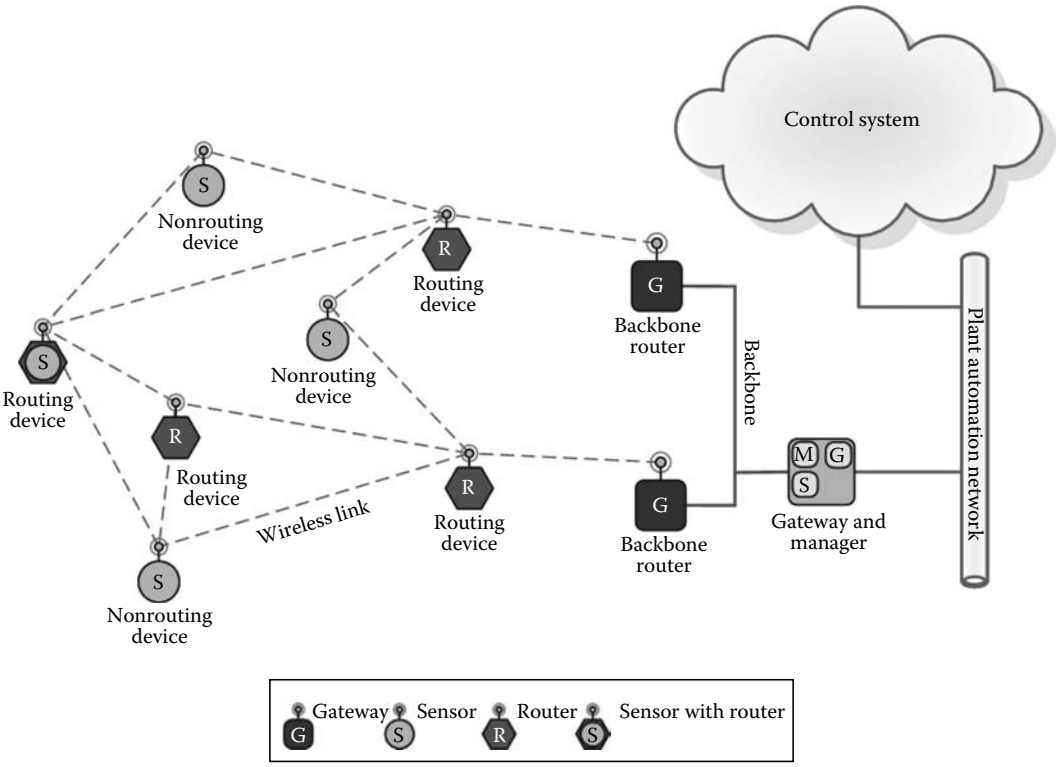


FIGURE 27.11 ISA100.11a network.

that UDP is an unreliable delivery service. Delivery reliability for a DLL subnet is handled in the DLL, whereas reliable delivery on the backbone network is handled by the backbone protocol, which is not within scope of the standard. The transport layer also contains options for securing the delivery of packets, e.g., encryption, authentication, integrity, etc.

27.5.2.3.2 Basic Functionality

As WirelessHART, ISA100.11a is a TDMA, frequency hopping, mesh network. In addition, it supports the IPv6 protocol, and the UDP transport protocol (through the use of 6loWPAN). Thus, all devices in an ISA100.11a network have an IPv6 address, which makes them globally visible within the plant.

In an ISA100.11a network, there are several different kinds of devices, as it can be seen in Figure 27.11. ISA100.11a supports both "routing" and "nonrouting" devices, as well as a special kind of routing devices, "backbone" routers, which utilize the backbone in order to transport messages within, or between, DLL subnets.

The DLL provides time synchronization, frequency hopping, mesh networking, routing, within a DLL subnet. A "time slot" in ISA100.11a can vary between 10 and 12 ms. This flexibility allows the use of "duo-cast," which is a procedure in which each message is sent to two receivers, who both answer with an ACK on successful reception. Even if one transmission fails, the other might succeed, which increases the communication reliability of the network.

Frequency hopping can be either "slotted," i.e., the frequency is changed for every slot or "slow" i.e., the frequency is only changed when several sequential slots have passed. Slow hopping relaxes the time synchronization requirements for devices, allowing low-cost clocks and crystals to be used.

ISA100.11a DLL subnets can have many topological shapes: mesh, star, or hybrid (a mix of both), and the network as a whole also incorporates the use of a backbone network. The Gateway/Manager

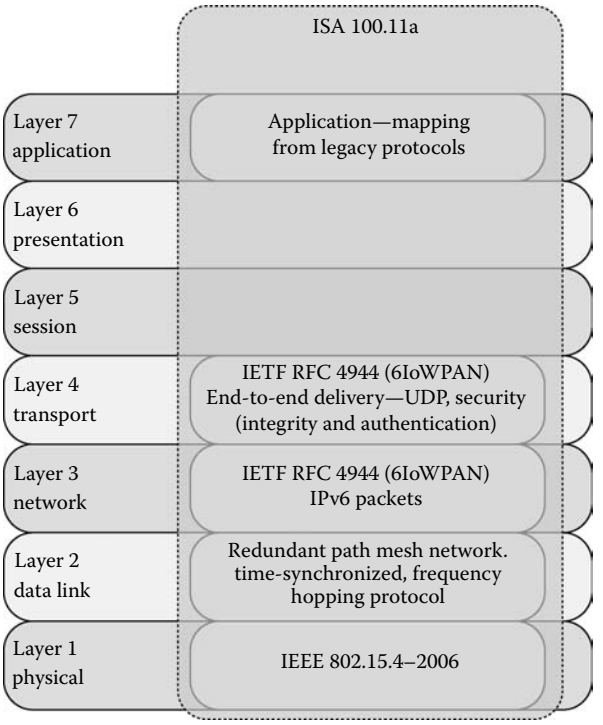


FIGURE 27.12 ISA100.11a protocol stack.

device can be connected either directly to the DLL subnet via a radio or through backbone routers, in which case it is located somewhere on the backbone network. Routing in DLL subnets is handled either through the use of graph routing (preallocated redundant paths) or source routing (ad hoc-created nonredundant paths), but routing on the backbone network is not within the scope of the standard.

27.5.2.3.3 Security

ISA100.11a provides a two layer security measure; single-hop and end-to-end security. The single-hop security is handled in the MAC sublayer (DLL), and the end-to-end security is handled in the transport layer.

Both security layers use shared-secret symmetric keys (it is optional to use asymmetric keys) and AES-128 in order to provide encryption, authentication, and integrity checking. Payload encryption is optional in ISA100.11a. The ISA100.11a specification is designed with future "upgrades" of the security specification in mind.

27.5.3 Why Is ZigBee Not Enough?

Since any problems with the equipment translate to economical loss for an industrial user, reliability is a primary concern for these users. Hence, parameters like network robustness, reliable message delivery, authentication, and integrity are all important for industrial use. Moreover, the threat of industrial espionage advances the requirement for encryption to hide information that can reveal anything about the production in the plant to competitors.

One of the loudest arguments against ZigBee has been the lack of industrial-grade robustness. First of all, there is no frequency diversity since the entire network shares the same static channel, making it highly susceptible to both unintended and intended jamming. Moreover, the static channel will

also increase interference for other systems like wireless-LAN, and increase delay as the network size grows and collisions force retransmissions.

Secondly, there is no path diversity meaning that in case a link is broken; a new path from source to destination must be set up. This increases both delay and overhead, and eventually will consume all bandwidth available.

Battery operation for routers with many peers is not realistic, since the CSMA/CA forces it to keep its receiver on for a large part of the frame. Keep in mind that industrial users require several years of battery life, since frequent battery change on thousands of sensors is not an option.

Security in ZigBee can to a great extent be set up to meet the requirements from industrial users, although care must be taken to use equipment from vendors which support the necessary security mechanisms. A good guideline for ZigBee security is found in Ref. [8].

In summary, the harsh operating environment, battery operation, and lifetime requirement render ZigBee a nonviable option for the industrial automation application. WirelessHART on the other hand is very suitable as it is designed to be a very robust communications technology in the presence of interference and fading. In addition, WirelessHART has features that allow it to efficiently coexist in a "crowded" frequency band, i.e., it is a "nice neighbor."

27.6 Low-Power Design

In this chapter, we dive more deeply into the issues related to low-power design of hardware, software, and the communication protocol.

27.6.1 Basic Principle

The aim is to use the available resources within the acceptable limits of the specification and never have anything powered up, if it does not have to be. The task reduces to switching units, such as the sensor, CPU, and transceiver on and off with the right timing. Assume a node needs to wake up at regular intervals and transmit its sensor value if it differs from the last value by more than a predetermined amount. After the value has been sent over the radio channel, the unit waits for an acknowledge message indicating that the packet has been correctly received.

The required behavior of the node is best explained using what is known as a state diagram: a schematic representation of the state the node is in, the events that may cause it to move from one state to another, and the actions associated with each state transition.

Figure 27.13 illustrates a typical high-level state diagram for a sensor node, in which the "Measure" and "Send" states together form an operational mode. The transition from active to sleep is managed by a sleep–wake-up scheme, which has been derived from the measurement duty cycle requirement.

A natural prerequisite to implement an efficient sleep–wake-up scheme is that the active/sleep state of various system components can be controlled externally. If and how this is accomplished can vary greatly between components and can have quite an impact on the duty cycle of the device.

Wireless Vibration Monitoring Case Example

The vibration sensor is build around the following controllable components:

- *CPU*: the microcontroller has several controllable low-power modes.
- *External real-time clock* (*RTC*): the clock can be powered on and off, as well as controlled using its "normal" software interface.
- *Radio chip*: the radio is a black-box that only provides on/off control, as well as a reset possibility.
- *Analog filter electronics*: these filters can be powered on and off.

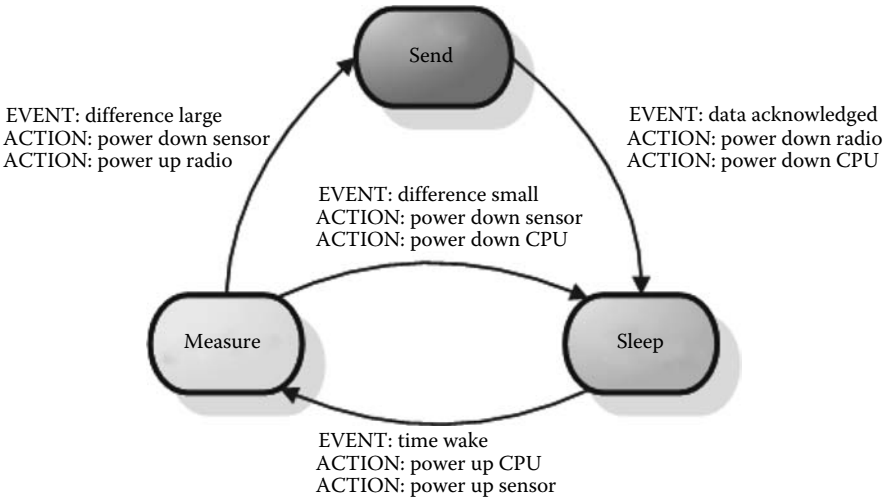


FIGURE 27.13 Sensor node state diagram. (From Aakvaag, N. and Frey, J.-E., *ABB Rev.*, 2, 2006. With permission.)

The state diagram shown above is rather simplistic as it only shows the "normal" measurement use-case. In real life, many additional and alternative use-cases need to be considered. As the following sections will show, this simple state chart can become quite complex to design.

27.6.2 Sleep Modes

The first parameter to consider is the power consumption in normal operating mode (i.e., active mode) for the CPU, sensor, radio transceiver, and possibly other elements such as external memory and peripherals. However, since the sensors spend most of their time "sleeping," it is important that the consumption during sleep mode is low. Often, we can switch power to the sensor and transceiver off completely. However, the CPU will need some form of sleep mode from which it can be woken. The consumption in this sleep mode is absolutely crucial for the overall power budget.

Additionally, one must ensure that all the necessary elements can be controlled by the CPU. It is the master in the system and needs to have complete control over the other functional blocks. However, the possibility to exert this control often varies in the different power modes supported by the CPU. The lowest power-saving mode typically shuts down (stops) the CPU, which will not allow it to control anything.

Wireless Vibration Monitoring Case Example

The CPU, a Texas Instruments MSP430 variant, used in the vibration monitor device supports five low-power modes. The CPU can be operated at voltage levels ranging from 1.8 to 3.6 V. When operating at a 3.0 V supply level, the "lowest" power-saving mode (the CPU is practically turned off) typically requires 0.1 μA, in "standby" mode (next lowest) it requires 1.6 μA, and in "active"mode it requires 400 μA. Looking at these numbers, it is obviously most desirable to use the "off" mode during sleep time, as it has a 16X lower power consumption compared to the standby mode. The drawback with the off mode is that the CPU can only wake up from external stimuli (i.e., an externally generated interrupt), which is not the case in the standby mode (or any of the other low-power modes).

One aspect that is often overlooked is the time the elements need to turn on and off. For example, the transceiver will need some minimum time for its oscillators to stabilize. While they wait, both the transceiver and the CPU burn power. This therefore needs to be minimized. The same obviously holds true for both the CPU and the sensor.

Wireless Vibration Monitoring Case Example

The sensor node uses analog electronics components to filter the "raw" vibration signal into a desired format. The CPU has control over the activation/deactivation of this analog part, and only activates it for the duration of the measurement. The analog part has a fairly long settling time, ~15 s, which forces the node to wait until a usable signal is available for processing (fortunately, the CPU can spend this time in the standby low-power mode).

Another very important settling time is the time it takes for the CPU to become fully operational when waking up from a power save mode. For the MSP430 variant used, this time is only 6 μs from both the off mode and the standby mode. But, the off mode has the added benefit of allowing us to further decrease power consumption by shutting off the oscillator signal from the RTC to the CPU (which is used to generate the CPU clock). However, without the oscillator signal, the CPU will default to an inaccurate, internal, RC-generated, lower frequency clock. Thus, when waking up from the off mode, the RTC oscillator signal has to be activated before the CPU will stabilize, which adds to the 6 μs wake-up time.

Timers are an integral part of any sleep–wake-up scheme. Generally, ultralow power CPUs only have 8 or 16 bit resolution timers, which, when used to generate interrupts at scheduled timeouts, have fairly "short" maximum intervals (in the order of a few seconds). If a longer sleep interval is desired, the timer is forced to wake up (i.e., the CPU becomes active), reset its counter, and go to sleep several times during that interval. Furthermore, in some power-saving modes (as described in the previous example), the CPU shuts down its internal clock, which disallows the use of the internal timers. Thus, if internal timers are the choice for the wake-up scheme, some power-saving modes will not be usable.

Wireless Vibration Monitoring Case Example

The duty cycle of the vibration monitoring device is 2 weeks. The MSP430 CPU used is driven by a 32 kHz crystal, and only provides 16 bit timers, which equates to a maximum interval of only 2 s. This would require the timer interrupt to (a) activate, (b) reset the timer counter, and (c) decrement the total sleeping time, a total of 604,800 times (14 days = 1, 209, 600 s and a 2 s timeout) during the 2 week interval! In combination with the desire to use the lowest power-saving mode (off mode) of the CPU, the solution was to choose an external timer, i.e., an RTC, which supports very long timeout intervals (actually years), and allows the CPU to stay in the "off" power-saving mode during the sleep time.

27.6.3 HMI

As noted in previous sections, a local HMI is commonly required to check the status, calibrate, and maybe even instruct the device to perform a measurement. From a low-power design point of view, this represents additional components that consume power. Depending on the requirements on packaging or the available power budget, several alternatives are used in the industry ranging from simple LEDs and buttons to LCDs. When designing the HMI functionality, great care must be taken how the HMI is used. As battery-powered systems cannot afford to constantly have the HMI active, it

must therefore be switched on and off at appropriate times, while maintaining a user friendly interface (e.g., that responds in a timely manner according to your expectations).

When the HMI is used to asynchronously wake up a device and maybe issue a request to perform a measurement, the normal duty cycle is changed, which will result in a different power consumption of the device. The update rate of an HMI is typically a lot faster than the normal duty cycle of the WSN devices (since it is often used to get a "quick snapshot" of the process). If the HMI is used extensively, not only will the HMI consume power, but the device will also run according to an entirely different, from the planned, duty cycle. This could result in early depletion of batteries, or even peak power consumption that exceeds the bounds of the available power.

Wireless Vibration Monitoring Case Example

The vibration monitoring device has an HMI consisting of two LEDs and one button. The LEDs are used to indicate activity, such as powering on/off, status (sleeping or active), and transmitting data. The button is used to activate or deactivate the device, and force it to start a measurement, and consequently transmit the data to its destination.

The device uses low-power LEDs, which have a current consumption of 2–4 mA when lit (LEDs are normally specified for a 4–20 mA consumption!); the exact consumption depends on the supply voltage level (2.7–3.6 V). Compared to the current consumption of the CPU in active mode, 400 μA, this is a very high consumption.

To further lower the power consumption of the HMI, it is only activated when a human is interacting with the device. Although it makes no sense to blink an LED if no one is watching, there are situations when this could be necessary, e.g., if the sensor is in a hard to reach place, but can be inspected visually. The button is used to interact with the device, and the response is always indicated by blinking the LEDs in different patterns. This pattern, e.g., short blinks, long blinks, etc., indicates the internal state of the device (sleeping, measuring, sending, etc.). In order to even further decrease the power consumption of the LEDS, the digital I/O pins used to control them are set to an inactive state when they are inactive.

It is important to note that an HMI does not necessarily have to be a physical component within the device itself. For example, a hand-held unit could communicate (wired or wirelessly) with the device and request data outside of the normal duty cycle. The same goes for a remote terminal connected to the network, which would cause additional traffic routed through the network. This means that in order to make any guarantees in terms of battery lifetime, the topology and usage scenarios need to be clear for the entire network.

27.6.4 Energy Efficient Protocols

In addition to utilizing low-power electronics and a clever software architecture (i.e., sleep–wake-up scheme), the communication protocol has a major impact on the final power consumption of the system. The communication protocol prescribes when the system needs to wake up, how it synchronizes, how long it needs to stay on, etc. Relevant protocol design choices from a power consumption point of view include:

- Media access scheme: In a contention-free media access scheme, e.g., TDMA, collisions are minimized, thus requiring less active time to send the data. The reverse is true for contention-based media access such as CSMA, where the node first has to listen before it can send its data (which degrades rapidly when there is lots of communication).
- Topology: The topology not only affects the timeliness of message delivery, but also the power consumption. In a multi-hop system, a node may have to wake up to forward data

from other nodes, which is not the case for a single-hop system. In addition, intermediate nodes do not know when they may be called upon to route packets for others. This is what motivates the Hybrid Mesh Topology with mains-powered router nodes shown in Figure 27.2.

- Data package size: The size of the payload, header, CRC, etc., all have a direct impact on how long the radio needs to be turned on to send its data. This is regardless of what media access scheme is employed. When designing the protocol, a trade-off has to be made between throughput, latency, and power consumption; and at least one of them has to suffer.
- Security: It is important that the data are secured from intentional corruption (i.e., integrity), and that it is sent from a valid device (i.e., authentication). Security adds overhead to the packets and to the computation performed at the devices (i.e., to check the integrity and authenticity). This, in turn, possibly leads to the need of specific hardware (i.e., encryption support) in order to achieve the required communication timing requirements.

Details in the communication protocol thus dictate the lower bounds of the consumption. Some communication protocols are notoriously inefficient and no smart embedded programming in the world will help get the consumption down to an acceptable level. Others are designed to give low consumption without unduly compromising communication performance. The WISA [9] platform is one such low-power protocol. The high performance can be attributed to two reasons: it is single hop and it uses time division multiplexing. The former avoids delays in intermediate nodes. The latter guarantees that a node will be alone on the channel, i.e., there will be no collisions. WISA is designed for manufacturing control applications, which require high node density and deterministic response times, but have rather relaxed requirements on throughput and range.

The recently developed WirelessHART specification with the underlying 802.15.4 protocol is more general, but will have lower communication performance. It specifies multi-hop, where a message can use several radio hops to get to its destination. This introduces uncertainty in the delivery time of a message, but extends the range while keeping power consumption acceptable.

In short, the WISA protocol is well adapted to the requirements of discrete manufacturing, while WirelessHART is ideally suited for asset monitoring applications.

Wireless Vibration Monitoring Case Example

In the WSN of the reference case, all nodes have the capability of routing traffic. The Network manager continuously changes communication links in the mesh network, in order to cope with fading and interference, as well as trying to evenly spread traffic over all nodes (thus evenly spreading the power consumption between the nodes).

Furthermore, the TSMP solution provided by DUST Networks allows the designation of leaf nodes, i.e., nodes that will never route any traffic. This provides the possibility to form a mesh backbone network consisting of pure routing nodes (i.e., without vibration measurement capability) and to have all sensor nodes configured as leaf nodes, which will further decrease their energy consumption.

27.7 Packaging

27.7.1 IP Rating

The international standard IEC 60529 [12] classifies the degrees of protection of an electric system against solid and/or liquid. The rating consists of the letters IP (International Protection rating)

followed by two digits and an optional letter. Certain IP codes require that the device is completely sealed, which might cause problems for the antenna (detuning) and battery design. A completely sealed device does not permit an easy access for battery change. The sealing could also force the use of an internal antenna, i.e., inside the housing.

27.7.2 Hazardous Environments (EX)

When designing devices intended for use in potentially explosive atmospheres, special measures need to be taken to prevent or minimize the risk of explosion. This means that you must consider

1. Every possible electrical or nonelectrical source of ignition
2. All potentially hazardous environments where the device could be deployed
3. The different ways the device could be deployed and the technical ability of the person using the device

To design and develop devices for hazardous environment requires long experience in the field as well certification by a notified body. The overview provided in this chapter should thus not be considered a cook book for safe designs.

Conformance to a number of standards is required to achieve EX certification. The IECEx* Scheme [10] was created to facilitate international trade in equipment and services for use in explosive atmospheres, while maintaining the required level of safety. The IECEx Scheme is based on the use of specific international IEC Standards for type of protection of EX equipment. Since July 2003, EX products that are placed on the European market must be certified to the ATEX† directive (ATEX 94/9/EC) [11]. This involves the testing and assessment of such products to the latest ATEX standards.

To achieve an EX safe design, a number of principle protection concepts are applied, depending on the applications and target hazardous area (zones).

- Nonsparking devices: Typically applied to transformers, motors, cells, batteries, etc., but not appropriate for equipment or components containing semiconductors. Protection relies upon a dust/water tight enclosure to avoid tracking across live circuits.
- Keep flammable gas out: Encapsulation (enclosing circuit in potting compound), oil immersion, or pressurized enclosures to keep the potentially explosive atmosphere away from the source of ignition. The downside of this approach is that it adds significant cost and weight to your design and sacrifices accessibility and serviceability of the device.
- Limit energy of sparks and surface temperatures: Protection relies upon an electronic design that limits the emitted/transferred power of the device and prevents hot surfaces above the given temperature class. This approach is commonly called Intrinsic Safety.

An IS design has built-in measures to prevent excessive heat buildup and the danger of an electric spark. To achieve this, you need to put limitations on voltage, current, capacitance, and inductance

* International Electrotechnical Commission for Certification to Standards Relating to Equipment for use in Explosive Atmospheres.
† From the French-ATmospheres EXplosibles.

to ensure that the available energy within the device is below the minimum ignition energy of the potentially explosive atmosphere.

A practical implication of this is that you need to limit the maximum amount of energy supplied by the power supply, as well as dividing your design into clearly isolated blocks, and limiting the energy transfer between these blocks. The interfaces between the system blocks thus constitute the majority of the design work and also have the most dramatic impact on other design aspects.

Additional components are typically required to limit discharge energy or isolate blocks, e.g., resistors, fuses, and Zener diodes. These additional components tend to increase the overall power consumption of the device as you will experience losses when traversing through the different blocks of your design.

In addition to providing means for limiting the energy, you also need to provide some guarantees that these protection and power limiting components will not fail. When designing an IS circuit, all possible scenarios for connections of components need to be considered when calculating the safety circuit parameters. Defining clear interfaces helps determine worst case failure mode and devise possible solutions to handle these failures. Typically duplication or triplication of components is required, as well as using the components below their specified ratings and observing sufficient creepage and clearance distances. This further lowers the efficiency of the device and can impact the form factor of the device and require additional redesigns.

Since a component with a high surface temperature could cause an ignition of the gas, the maximum temperature of any component on a circuit board under fault conditions must be considered. Limiting the energy alone does not automatically result in a low component surface temperature, especially when considering small components. This means that you need to take great care when selecting components and ensure that the heat dissipation is satisfactory to avoid an excessive heat buildup. Usually, this is solved by choosing sufficiently large components, which again can impact the overall form factor of the electronics and lead to space problems.

Wireless Vibration Monitoring Case Example

A battery with IS approval was selected. The battery is connected to the PCB of the device via an infallible current limiting resistor. This will ensure that the maximum power loss in any component on the PCB is limited. The temperature rise can then be calculated for each component on the PCB.

In addition, the resistor was carefully chosen with a specified temperature rise that will ensure that the maximum power dissipation in the resistor upon a short will not result in an unacceptable temperature rise in the resistor.

Since the battery voltage level will vary with the surrounding temperature, we might not have enough energy to operate the radio circuit under all conditions (especially during start-up). To solve this, a capacitor bank was connected across the power supply. The capacitor bank stores just enough energy required for the radio transmission. The capacitor bank was carefully designed to comply with relevant safety standards (Figure 27.14).

27.7.3 Other Environmental Considerations

In addition to the IP and EX environments described above, most industrial environments are also subject to vibration and extreme temperatures.

In an environment where the device is subject to severe continuous or occasional vibrations (shock), there is a risk that internal components will shake loose due to these vibrations. Therefore, as a protection, a device can be filled with a shock absorbing material. Drawbacks with this solution are that the filling materials usually has a high cost, and can force a design solution where the battery is also encased in the material, which make battery changes impossible.

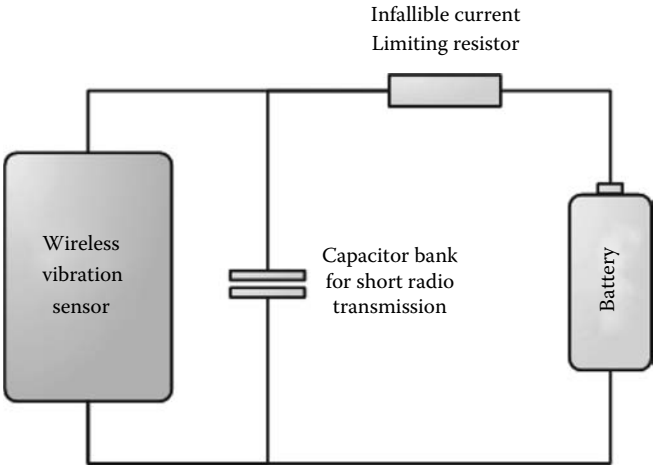


FIGURE 27.14 Battery design.

Depending on the specified tolerance of the device components, a high or low temperature can force the device to have a thermally insulating housing, or some kind of cooling mechanism to protect its internal components. Thermal insulation housing consists of materials that do not transfer heat quickly, in order to separate and insulate the sensitive internal components from the external temperature. It is also possible to have a thermally insulating barrier, a non-heat conducting material positioned between the hot/cold part and the critical part of the device.

There are different kinds of cooling mechanisms available: active, such as fans, and passive, such as heat sinks and heat pipes. A problem with active cooling components is that they also wear out and could pose problems if they fail, e.g., overheating. This would require maintenance of the cooling mechanism on the device, possibly outside the normal maintenance schedule, which is too costly. Thus, active cooling is not a desirable solution.

27.7.4 Form Factor

The form and shape of an embedded device is very important. It depends on various requirements, such as

- The environment where the device will be located, which usually is very close to the process it will monitor
- Possible reuse of legacy housing
- Mounting requirements
- Design guidelines of the device manufacturer

The housing material of the device is also a very important part of the form factor. As mentioned in the previous subsection, there might be a requirement for the material to be thermally insulating or cooling, or just that it should be rugged, which also affects the physical dimensions of the device, e.g., weight difference between a metal and a plastic housing. Casting or filling also affects the weight of the device, and can also make component replacement or upgrading impossible, e.g., battery replacement.

The power source, e.g., a battery or capacitor, will affect both the weight and physical shape of the device, and thus has a major influence on the design. Batteries (and capacitors) have a correlation between stored power and physical dimensions, i.e., larger batteries can store more energy. If the form factor forces a certain maximum physical size for the power source, the device's life length will

be effectively limited by that size. Low-power design will push the limits of the device's life length based on the resulting power source.

The resulting physical form can impose limitations and requirements on the design of the PCB and positioning of electronics components, which could have performance and maintenance effects. For example, the form factor could force the antenna to be mounted at a nonoptimal position (i.e., from a radio communication point of view), which could cause degraded communications performance. It is also possible that when selecting certain components, or a combination of components, the whole PCB must be redesigned in order to fit the components onto it in conjunction with having it fit within the device housing.

Wireless Vibration Monitoring Case Example

The physical form and shape of the vibration monitoring device is very much dictated by the dimensions of the battery, which is selected based on the life length requirements on the device. This in turn restricts the shape of the PCB and thus the area available for mounting electrical components. The result is a cramped area, which has forced a solution that uses both sides of the PCB for component mounting (which has a slightly higher cost).

The weight of the device is also an important requirement. It should be as low as possible because the device weight directly influences the quality of the vibration measurement (a heavy device is harder to move, i.e., vibrate). A related requirement is the size of the mounting area for the device; this should be as small as possible while still be large enough to provide solid mounting, which also affects the quality of the vibration readings. Finally, a more esthetic requirement is that the device should have a similar look and feel as a rugged industrial accelerometer. Figure 27.15 shows the sensor prototype design.

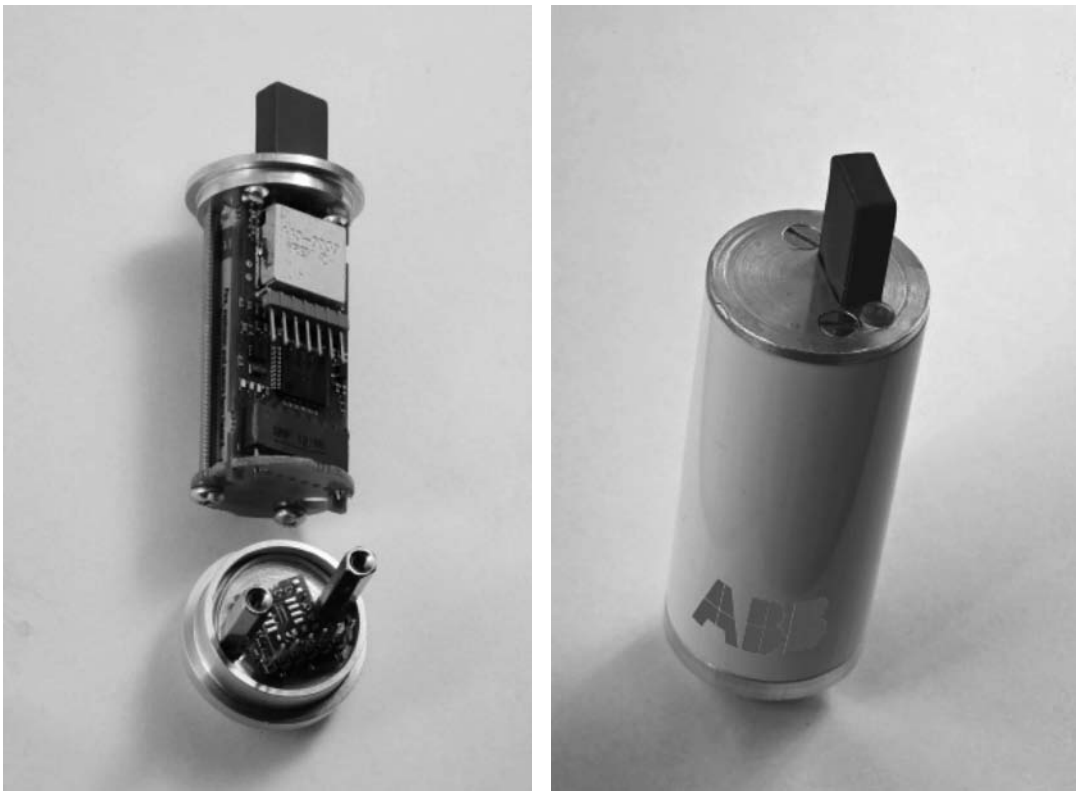


FIGURE 27.15 Sensor prototype design.

TABLE 27.4 Modularity Trade-Off Examples

Aspect	Pros	Cons
Reuse elements	Minimized development effort due to reuse of HW/SW design (and code)	Lower performance (footprint, response time, power consumption, etc.) due to increased overhead
Risk reduction	Design based on well-proven and tested components	Integration of several components can result in unclear/untested behavior
Migration	Minimized effort to migrate to other (newer) SW/HW versions	Lower performance due to "wrappers" needed to maintain system interfaces
Cost reduction	Lower cost due to second source and larger volumes ("standardized" components)	Added cost due to additional resources required to support the standardized components

27.8 Modularity

The main driving force behind a modular design is improved maintainability, i.e., how easily we can modify the device in order to fix bugs, implement new functionality, upgrade to another SW/HW, version, etc. This is achieved by a clear separation of system features into modules or components that overlap in functionality as little as possible. These HW/SW modules are integrated into the device via clearly defined interfaces, which provide an abstraction of the module, i.e., a separation of the external communication with the module from its internal operations. This means that (in theory) you are able to modify the module internally, or even replace it with another similar module, without affecting how the rest of the device is operating or interacting with the module.

However, modularity comes at a cost as it places restrictions on the design, and care must be taken to ensure that the interfaces between modules are sufficiently general, while adhering to the nonfunctional requirements (response time, footprint, power consumption, etc.). Table 27.4 below illustrates some typical aspects or goals of a modular design and their trade-offs.

As modularity does come at some cost, we have to balance the need for a highly modular design with, on one hand the nonfunctional requirements, and on the other hand the products envisioned life cycle. There is no need to pay the price for a highly modular system if you are going to change it once every 10 years. However, it can be quite tricky to foresee the maintenance need of industrial products very accurately as it depends on large variety of aspects such as new user requirements, deployment in unforeseen applications, and availability of the COTS components that make up the design. Some form of modular design is therefore generally speaking a well-invested development effort.

27.8.1 Levels of Modularity

What does a modular design really mean? Are not all modern embedded systems modular in some sense, since they are built on standardized electronic components that you purchase and integrate on a PCB? The answer is yes and no. Yes, the components are standardized and can be replaced by another brand or type, but this does not automatically equate to a design that enables easy modification, extensions, migration, or reuse of parts of the design.

There is always a value of standardizing on a given set of low-level components, but to achieve a higher level of maintainability, you often need to raise the level of abstraction and divide the device into a set of core system features that overlap in functionality as little as possible.

27.8.1.1 HW vs. SW Components

Standardizing on a given (set of) processors (CPU, MCU, DSP) of the same architecture has benefits in terms of reduced training cost of engineers and facilitates porting of software. Reuse of printed

circuit boards, either actual hardware or of layout with design modifications, has a direct saving in production cost, testing, and competence buildup. Reuse of mechanical solutions such as the packaging of the device often has large direct cost savings due to the elimination of design costs and cost of tooling. However, you need to balance the cost savings associated with utilizing a common housing, with the design restrictions, this will put on other parts of the design (form factor, antenna placement/design, etc.).

Reuse of software components is advantageous under the condition that the sum of costs related to purchasing the code (internal or external), competence buildup, modification of components, integration of components, and testing is less than the cost of designing a component from scratch. Experience has shown that this is not easy unless other elements of the development organization are in place such as clear organizational roles, requirement management, product platform architecture, and design, etc.

27.8.1.2 Single vs. Multiple Chips

One of the major design choices that must be made is whether to integrate functionality in one hardware chip or to have it distributed into different chips. In certain cases, there is no choice, the desired functionality is only supplied as a separate chip, i.e., as a COTS component, while in other cases the component can be designed in-house and suitably located. Typical functionality and subsystems available in a WSN device are CPU, radio communication, sensor, signal processing (filters, DSPs, etc.), RTC, external memory (RAM, ROM, Flash, etc.), HMI (LEDs, buttons, etc.), and an AD converter. Many of these subsystems could be integrated into one chip, e.g., the CPU could contain almost everything.

A typically design choice for a WSN sensor device is whether to separate the radio communication stack and the application into different chips (see Figure 27.16). To conclude on a suitable solution, a trade-off between low-level control, form factor, cost, overhead, and performance has to be made.

Figure 27.16 above shows a very clean separation between the application and the communication stack. In reality, this separation is often far from clear. A typical example is what a stack supplier considers being part of the communication stack and what is considered to be the "application" (and thus excludes from the supplied stack offering).

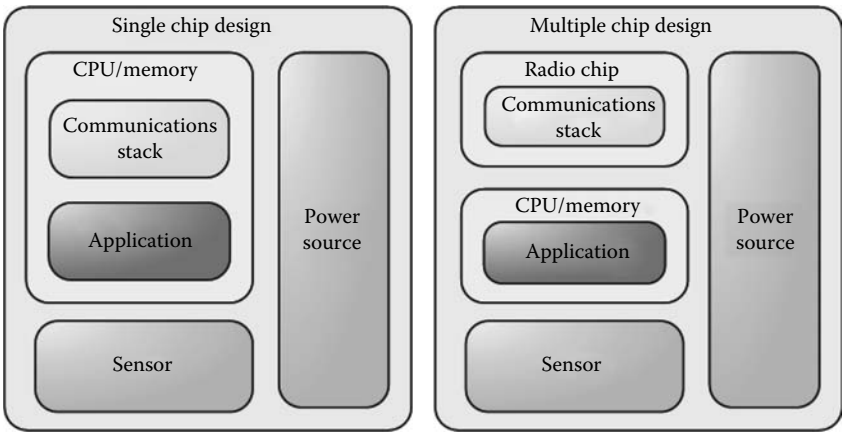


FIGURE 27.16 Example of single vs. multi-chip solution.

Wireless Vibration Monitoring Case Example

In the reference case, the chosen communication solution, i.e., DUST Networks SmartMesh XD chip, provides a complete SoC solution, where both the hardware and software are hidden within the chip.

However, a small portion of the communications interface resides outside the SoC component, possibly in the application. This is functionality which handles packet preparation, such as byte stuffing and Frame check sequence (FCS) calculation for packets which will be transmitted, and the corresponding operations for received packets.

This means that we need to implement communication-related functionality in the application, which normally would be considered to be part of the protocol stack.

Modularity can also be achieved at a lower level. For instance, the communication protocol can be seen as consisting of several blocks, known as the OSI layers. Given a healthy design procedure, one may be able to exchange a single layer with one from a different source. Obviously, the more the code is split up, the more modular it becomes. At the same time, the risk of suboptimizing increases, i.e., the modules are optimized (with respect to power, performance, code size, etc.) individually, but this does not necessarily give a globally optimum solution.

27.8.1.3 Lifetime

Industrial automation devices have a life expectancy that could be measured in 10ths of years. This puts a lot of demand on a modular and flexible design, which is easily maintained. Even if the device functionality would not change over its lifetime (which is highly unlikely), we still need to cope with availability of components. For certain parts of a WSN device, where the technology changes more rapidly, this is especially critical. For example, the sensor part (see Figure 27.1) evolves quite slowly, whereas the radio and power source technologies evolve very quickly. This fact warrants a high degree of modularity around the radio solution, something that is not always advantageous from a performance point of view.

27.8.2 Interfaces

The key to achieving a good modular design, and thus being able to reuse components, are the component interfaces. The interface provides an abstraction of the component, i.e., a separation of the external communication with the component from its internal operations. This abstraction enables the internal modification of a component without affecting how the rest of the device is operating or interacting with the component.

A well-defined interface allows for easy maintenance and reuse of the component, but since interfaces are a form of indirection, some additional overhead and decreased performance is incurred vs. direct communication. A modular design can thus introduce extra overhead in the sense of protocols needed to communicate between the different components (using the interfaces), which leads to increased power consumption and can result in bottlenecks in the system. The reason is that the system can only operate at the "speed" of its slowest component, at least if that component is part of the central functionality of the system.

Reusing and exchanging integrated circuits (ICs) requires that their interfaces are compatible with each other. This is commonly called pin compatibility, and comprises

- Functional compatibility: implies that two ICs have the same functions (inputs, outputs, power supply, ground, etc.), assigned to the same pins
- Mechanical compatibility: ensures that ICs can be inserted into the same socket or soldered to the same footprint
- Electrical compatibility: implies that the components work with the same supply and signaling voltage levels

Pin compatibility can play a major role when choosing components as it offers a degree of freedom in terms of migrating from one HW component to another without requiring a complete redesign of the PCB.

27.8.3 Subsystem Behavior

27.8.3.1 Local vs. Global Optimum

As discussed above, the sum of the parts can equal more than the individual parts together, at least from a product maintenance point of view. However, great care must be taken to ensure that sum of the parts also provides adequate performance (speed, power consumption, footprint, etc.). Even if each component has been optimized individually (for speed, power consumption, footprint, etc.), this does not guarantee that the final system behavior is optimal.

The principle operation of a WSN device can be described by the generic state machine depicted in Figure 27.13. This describes the overall behavior of the WSN device. The components that make up your device will in turn have their own state machines. The timing and interaction between the components thus is critical to arrive at an optimal overall system behavior. Care needs to be taken to ensure that a given functionality from one component is available when it is needed (according to the overall device state chart), or that enough memory or power is available to execute the requested operation. This can be difficult to achieve, especially if COTS components are utilized where you have little influence over the internal behavior of the component.

27.8.3.2 Control Freak

When reusing components, either developed in house or COTS, they will provide an interface and level of control that could be nonoptimal from a low-power design perspective. For example, if the communications stack is a black box COTS component, which only provides limited possibility of controlling its internal behavior, it could seriously affect the sleep–wake-up scheme, and thus the power consumption, of the device. How much control a component allows will thus dictate the low-power design, which we describe in detail in Section 27.6.

The interfaces are thus the Achilles' heel of a modular WSN design. As stated before, they need to be sufficiently general to enable efficient integration and porting, while not adding an excessive overhead. In addition, the interfaces also need to provide some control of the internal behavior of the component to be able to mediate between generality and performance. This goes against the basic principle of information hiding employed by component-based development, where internal state information is not shared and interaction is accomplished by exchanging messages-carrying data. However, in low-power WSN devices, where the system is a sleep most of the time, this "low-level" control is sometimes crucial to adhere to the sleep–wake-up schedule of the device.

27.8.3.3 Debugging a Black Box

Using black box components has the advantage that the functionality provided by the component is completely hidden. The developer thus does not need to bother about how the component works in detail, nor invest time and effort in testing and validating its functionality. In theory, this is true, but in addition to loss of control as described above, there is an additional risk that problems will appear when debugging the total system and the component functionality is part of the debug trace. As the name states, it is a black box, and there is no way to peer into the box to see what happens. This increases the difficulty of locating the bugs, i.e., determine if they are present inside the component or in other parts of the system. This problem is especially pronounced if the COTS component is itself in a "beta stage" (which is very common in early development phases of new technology) and bugs are certain to exist within it.

Wireless Vibration Monitoring Case Example

In the reference case, a black box communication solution was chosen, i.e., DUST Networks SmartMesh XD chip. It is actually a complete SoC solution, where both the hardware and software are hidden within the box. There is very limited control over how the communication is managed, e.g., the protocol, wake-up scheme for radio, etc.

A small portion of the communications interface resides outside the SoC component, possibly in the application, which handles packet preparation. This makes the integration between the SoC and the rest of the system very tight.

The advantage is that minimal design effort has to be spent on the communications protocol. But, the drawback is that debugging is very difficult because of the limited control and debugging support available through the provided interface.

27.9 Power Supply

The power supply is a central part of the WSN device; it is the "motor" that provides power to the various parts of the system. This motor can take many different shapes and forms and consequently provide varying degree of performance and lifetime. The general requirement is an energy autonomous system that requires little or no manual intervention (charging, maintenance, battery change, etc.) to operate throughout its target lifetime. This requires a thorough requirements analysis of the power demands and a clever design that is able to bridge power outages and handle variations in power consumption.

27.9.1 Requirements

27.9.1.1 Power Demands

Before deciding on a suitable power supply technology, the power demands of the system need to be clearly defined. The power demand is directly related to the sleep–wake-up scheme of the device as discussed in Section 27.6. It can thus be divided into a constant "housekeeping" part corresponding to the losses associated with the operation of sensor and electronics, and a time-varying power level that is related to the communication.

27.9.1.2 Energy Buffer

Regardless of the energy source employed, an internal energy storage is commonly used in low-power WSNs. This internal buffer enables the power supply to provide an even supply level in spite of a varying power inflow (e.g., when employing an energy-scavenging solution), and to handle intermittent peak consumptions (e.g., during start-up sequences). If an alternative power supply with a discontinuous power inflow is used, the buffer size will mainly be determined by the time characteristics of the power source. As such, the charging and discharging of the internal buffer needs to be accounted for in the sleep–wake-up scheme of the device.

Wireless Vibration Monitoring Case Example

As the wireless vibration sensor is rated for temperatures well below zero, the battery voltage level might decrease too much with the load. This in combination with the current limiting resistor (IS adaptation) might result in a situation where we do not have enough voltage to operate the radio (especially during start-up). To solve this, a capacitor bank was connected across the power supply, which stores just enough energy required for short radio transmissions. See Figure 27.13 for an overview of the battery design.

27.9.1.3 Lifetime

Typical quoted lifetime values of industrial WSN devices are often closely related to the lifetime of the power supply. This means not only that the components of the power supply should not fail within the specified time frame, but also that the available energy is enough to operate the device for at least the specified period.

If the power supply is based on a permanent storage such as a battery, the lifetime is highly dependent on how the device is used in the field, i.e., it is dependent on the size of the network, the duty cycle, the amount of data to transmit, and any asynchronous (nonscheduled) data access. Poor radio conditions will result in increased retransmits, which further decreases the actual possible operating time. So, unless you specify these conditions, quoting, e.g., 5 years lifetime of a battery operated device really does not say anything about how long you can run your device in the field.

To determine the size (Wh) of the permanent energy storage required to reach the desired lifetime, you also need to account for material aging, leakage, and self-discharge effects.

27.9.1.4 Form Factor

The form factor of the device is a critical parameter when designing WSN devices and the power supply often has the most dramatic impact on the final form and shape of the device. There must be some volume or area available in the WSN device, which is appropriate to mount the energy storage or energy converter. This volume/area has a direct relation to the required energy density of the storage, or power density if an energy converter is used. The often desired small size and weight of WSN devices is thus a clear limiting factor when determining the suitability of available power supply technologies.

27.9.2 Power Management

Depending on the availability, time behavior, and output characteristics of the source some kind of power management will be necessary as illustrated in Figure 27.17.

Apart from the primary source or energy converter, a power supply solution could involve an intermediate storage (energy buffer), a backup supply (e.g., a battery, which would also allow to keep the buffer size within reasonable limits), and a control unit in order to cope with I/O power fluctuations and downtime periods, which are typical for many alternative sources, but also for the power demand of an event-triggered sensor.

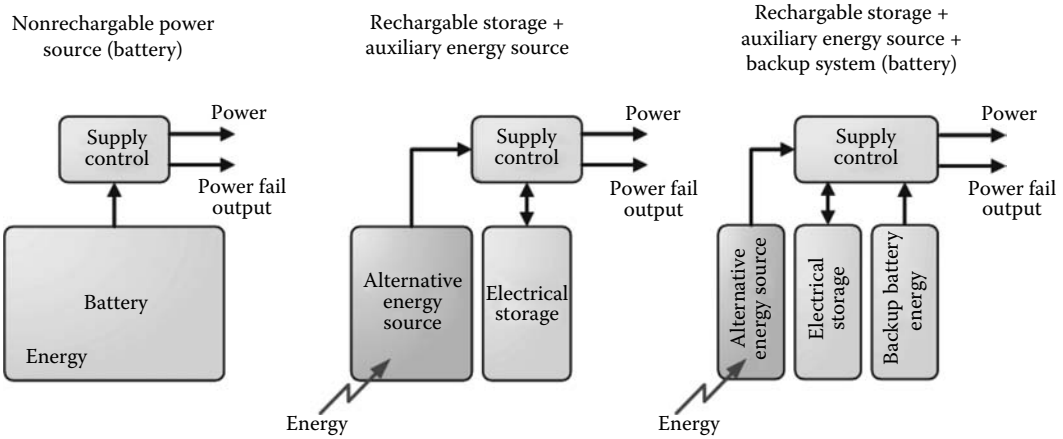


FIGURE 27.17 Power management.

Also, a DC–DC converter in the control unit might be necessary to adapt the voltage and/or current level of the source to that of the system and to increase the total efficiency. This is especially important at low available power levels. If, on the other hand, a battery is the main energy source, backup with alternative power can be used to extend its lifetime and reduce the need of service.

27.9.3 Sources and Technology

The energy required to operate the WSN device can be supplied in three principal manners. It can be stored in the device (energy storage), transmitted from a dedicated external source (energy transmission), or it could be based on converting waste energy from the environment, the monitored/controlled process or from some user action (energy conversions). The first category is by far the most prevalent method employed in WSN devices, while most research is being conducted in the energy conversion area. The approach of converting available energy, often labeled energy-scavenging or energy harvesting, holds the promise of energy autonomous devices with lifetime limited only by the component lifetime.

27.9.3.1 Energy Storage

If energy is stored in the device, great care needs to be taken when calculating the required energy needed to operate the device for the targeted lifetime (as discussed in Section 27.9.1). Energy storages can be divided into permanent storage and electrical buffer storage.

Permanent storages include:

- Primary batteries: Primary batteries remain the most common means to power WSN devices today. Although continuous improvement is being made to battery technology in terms of battery capacity and cost, the energy density has not really improved that much over the years. The practical energy density of primary batteries is around 1.2 Wh/cm^3. However, based on which technology you employ the energy density, open circuit voltage, self-discharge rates, and properties at high and low temperatures can vary greatly.
- Chemical storage/fuel cells: Combustible materials (chemical fuels) enable the storage of energy with very high densities, which can be converted into electric power by converters like a combination of a machine (motor, turbine) with a generator, or a fuel cell. Availability of fuel cells is currently limited to hydrogen, methane, or methanol, with achievable energy densities in the range of a little more than 2 Wh/cm^3.
- Heat storage: Heat storage systems make the use of the latent heat involved in melting or evaporation of so-called phase change materials (PCMs). Latent heat storages are for example used for the heating and cooling of buildings. Commercially applicable PCMs are available in a wide temperature range and with melting energies of up to 0.14 Wh/cm^3.

Depending on the required energy and power densities, electrical buffer storages can be used in many cases in order to buffer energy as described in Section 27.9.2. Two types of electrical buffer storages are commonly used:

- Accumulators: Accumulators are based on similar chemical principles as primary batteries, but they are designed for being recharged many times. Although performance is slowly approaching that of primary batteries, lower energy densities, significantly higher self-discharge rates, and the limited lifetime (around 3 years) must be taken into account.
- Capacitors: Double layer capacitors, also named supercaps, ultracaps, or gold caps, are special capacitors with very large capacitance and low resistance. As no chemical reactions are involved with the charging and discharging processes, rapid changes of the

stored charge, i.e., high currents, are possible. A major advantage over accumulators is longer lifetime (>10 years) and significantly higher number of load cycles. The technology is rapidly progressing and commercially available products provide energy densities up to 0.005 Wh/cm^3. Currently, they are often used in photovoltaic systems to bridge the day and night cycle of solar power.

Wireless Vibration Monitoring Case Example

The main requirement when choosing battery technology for the wireless vibration sensor was a high energy density (to comply with size/weight requirements). A primary battery based on a single lithium-thionyl chloride cell was chosen due to its high energy density, high nominal voltage, low self-discharge rate and overall good properties at both high and low temperatures.
However, this type of primary battery has two main drawbacks. Large batteries may pose safety problems (subject to transportation restrictions) and voltage delay due to passivation.

Due to the size requirements, the chosen cell was small enough not to be subject to any transportation restrictions.

Passivation on the other hand could still be a problem as it might lead to voltage delay. Voltage delay is the time lag that occurs between the application of a load on the cell and the voltage response. As we have an extremely low duty cycle in the wireless vibration application (each sensor node should perform one vibration measurement per 2 weeks), this behavior becomes particularly pronounced (since the passivation layer grows thicker). As described earlier, a capacitor bank was added in order to secure sufficient energy to operate the radio circuit despite a varying operating temperature (see Figure 27.14). This buffer also provides a remedy for potential passivation issues, as the initial energy will be drawn from the capacitor bank, thus eliminating any voltage delays.

27.9.3.2 Energy Conversion

Harvesting and converting waste energy from the environment certainly appears as an appealing solution, as the lifetime of the device would (in theory) be limited only by the component lifetime. However, this requires that the waste energy is readily available with a constant (or predictable) level. If this is not the case, large internal buffers might be required to guarantee reliable operation (which increases the size/weight/cost of the device). This also means that the applications in which the device can be deployed might be restricted, or that special considerations and/or measures need to be taken to guarantee some minimal level of power inflow (which makes device installation more complex and costly).

Energy conversions can be achieved in many ways, but the most common methods include:

- Photovoltaic cells: A wide range of photovoltaic systems are available on the market today based on diverse materials and technologies that have reached different maturity levels. Cell types also differ with respect to cost, electrical characteristics, efficiency, topology, thickness, shape, mechanical properties, and adaptation to various lightning conditions. Regardless of the varying energy conversion efficiency (5%–30%), the lightning conditions have by far the greatest impact on the feasibility of photovoltaic cells. In typical environments, the illumination density will vary within at least three orders of magnitude. For example, compare the illumination density of full sunlight (~100 mW/cm^2),

with normal office lighting (<1 mW/cm^2). This difference is even more pronounced by the nonlinearity of the cell output power, which results in reduced efficiencies at low illumination intensity.

- Thermoelectric converters (TECs): Utilize the temperature gradients to generate electric energy. If there is waste heat available which is normally conducted or radiated to the environment, it would be possible to connect a TEC to a hot surface, let part of the heat flow pass through it and generate the electrical energy, which could then be used to power the WSN device. In order to achieve sufficiently high efficiencies, TECs need to be appropriately designed and thermally and electrically matched to the respective application. Especially, in order to increase the heat flow (and with it the temperature difference across the module), it is important to make very good thermal contacts to the hot surface and to the environment at the same time. The latter requirement usually causes the use of air coolers or other heat spreading elements, which may be even much larger than the module itself.
- Mechanical converters: The most common type of mechanical converters is based on vibrations. In industrial applications, energy harvesting oscillators need to be adapted to low vibration amplitudes, high forces, and resonance frequencies of typically 50 to several hundred hertz, as this is the range of vibrations in machines. These conditions can be well adapted with piezoelectric converters, which can collect power levels of 10 mW up to several hundred milliwatt dependant on the existing acceleration amplitudes. However, sufficient vibration amplitudes cannot be generally assumed in all applications. Also, the mechanical system must be well tuned to the vibration frequencies (which may change both with the application and with time) in order to achieve high efficiency and power density. Design for broadband response will reduce the available output power amplitudes. One of the first commercial vibration energy harvesters for the supply of small wireless devices like sensors and transmitters was offered by a company called Perpetuum [13]. They offer inductive generators that deliver a few milliwatt and up to 3.3 V depending on the amplitude and frequencies of the vibration.

27.9.3.3 Energy Transmission

Another principle method of supplying energy to the WSN device is to transmit the power to the WSN device. This method thus requires some form of external power source that needs to be installed in the plant in "close" proximity of the WSN devices that are going to be supplied. The maximum distance between the power source and the WSN differs depending on the technology employed. Three main technologies can be conceived:

- Inductive Transmission: Currently used inductive power transmission systems are different with respect to range (wireless vs. contactless) and operation mode (continuous vs. pulse operation as often used with radio frequency identification [RFID]). A rather large variety of commercial solutions are available such as contactless rotary power and signal connectors, linear contactless solutions for rail and transport systems, and solutions for the supply of small consumer devices. In addition, RFID and transponder solutions are available from a large number of suppliers. ABB seems to be the only company offering a wireless "long range" (a couple of meters) inductive power supply system for sensors and actuators [9]. The system, called WISA-power, can deliver 6 mW of continuous power to the wireless sensors even under worst case conditions, and close to 100 mW to concentrator modules with larger receiver coils.
- Capacitive Transmission: Power transmission can also be achieved by means of high frequency electric fields. Contactless solutions for rotary signal transmission are available

on the market, but no "wireless" power solutions, as the available power levels are rather limited within the range of allowable field amplitudes.

- Optical Transmission: Optical power transmission is possible with fibers, a focused (laser) beam, or with diffuse lighting. In most applications, little would be gained by the replacement of wires by optical fibers. Focused beams are not applicable for the supply of many sensors dispersed throughout the automation facility/cell, where there often is no direct line of sight. An alternative could be the simultaneous supply of the sensors in an automation cell with a diffuse artificial lighting. However, reliability, cost effectiveness, and customer acceptance of such a solution make it a highly questionable approach. Even with such an approach, shadow effects cannot be excluded for general automation applications.

Although the list of power supply technologies above is by far complete, it should provide an idea of the various conceivable power supply solutions for WSN devices. Noticeable are the varying levels of energy/power density and characteristics of the power supply solutions, often highly dependent on the conditions under which they are employed. To choose a suitable technology, you thus need to fully understand the static and dynamic power demands of your device, as well as the applications and environmental conditions under which the device will operate. A more comprehensive overview of various solutions is given by Refs. [14,15].

References

1. http://en.wikipedia.org/
2. http://www.isa.org/, ISA Web site.
3. http://www.hartcomm2.org, HART Communication Foundation Web site.
4. http://www.dustnetworks.com, DUST Networks Web site.
5. http://www.zigbee.org, ZigBee Alliance Web site.
6. *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*, IEEE Computer Society, IEEE, 2006.
7. http://www.profibus.com/, International PROFIBUS Web site.
8. M. Ken, *Recommended Practices Guide For Securing ZigBee Wireless Networks in Process Control System Environments*, Lawrence Livermore National Laboratory, 2007.
9. D. Dzung, G. Scheible, J. Endresen, and J.-E. Frey, Unplugged but connected: Design and implementation of a truly-wireless real-time sensor/actuator interface, *IEEE Industrial Electronics Magazine*, 1(2), 25–34, Summer 2007.
10. http://www.iecex.com/index.htm, International Electrotechnical Commission for Certification to Standards Relating to Equipment for use in Explosive Atmospheres (IECEx Scheme).
11. http://ec.europa.eu/enterprise/atex/guide/, Guidelines on the application of directive 94/9/EC.
12. International Standard IEC 60529, Degrees of protection provided by enclosures (IP Code).
13. www.perpetuum.com, Perpetuum Ltd. Web site.
14. G. Scheible, Wireless energy autonomous systems: Industrial use? *Sensoren und Messysteme VDE/IEEE Conference*, Ludwigsburg, Germany, March 11–12, 2002.
15. J. A. Paradiso and T. Starner, Energy scavenging for mobile and wireless electronics, Pervasive Computing, *IEEE CS and IEEE ComSoc*, 4, 18–27, January–March 2005.
16. www.archrock.com, Arch Rock Web site.
17. www.omnexcontrols.com, Omnex Controls Web site.
18. www.savewithaccutech.com, Accutech Instrumentation Solutions Web site.
19. Wireless Automation, www.abb.com > ABB Product Guide > Low Voltage Products and Systems > Control Products > Wireless Devices.

28

Design and Implementation of a Truly-Wireless Real-Time Sensor/Actuator Interface for Discrete Manufacturing Automation

Guntram Scheible
ABB Stotz-Kontakt GmbH

Dacfey Dzung
ABB Switzerland Limited

Jan Endresen
ABB Corporate Research

Jan-Erik Frey
ABB Corporate Research

28.1 Introduction **28**-1
28.2 WISA Requirements and System Specifications **28**-3
28.3 Communication Subsystem Design **28**-6
Medium Access and Retransmission • Frequency Hopping • Antenna Diversity and Switching • Multicell Operation
28.4 Communication Subsystem Implementation **28**-9
Coexistence • Performance Measurements • Industrial Electromagnetic Interference • Interference from Other Communication Systems
28.5 Wireless Power Subsystem **28**-17
"Magnetic Supply": WISA-POWER • Resonant, Medium-Frequency Power Supply • Rotating Field • Omnidirectional Receiver Structure • Example Setup
28.6 Practical Application and Performance Comparison **28**-21
28.7 Summary **28**-26
References **28**-27

28.1 Introduction

Wireless communication is increasingly used in many industrial automation applications, which have many differing requirements [1]. The main automation domains are

- Process automation, typically dealing with continuous processes (e.g., the continuous or batch processing of liquids)
- Factory automation, where discrete, stepwise manufacturing dominates (e.g., mounting a product from many different parts)

Focusing on discrete manufacturing, again different application levels exist (see Figure 28.1). Consequently, there is a need for different wireless implementations to cover differing applications

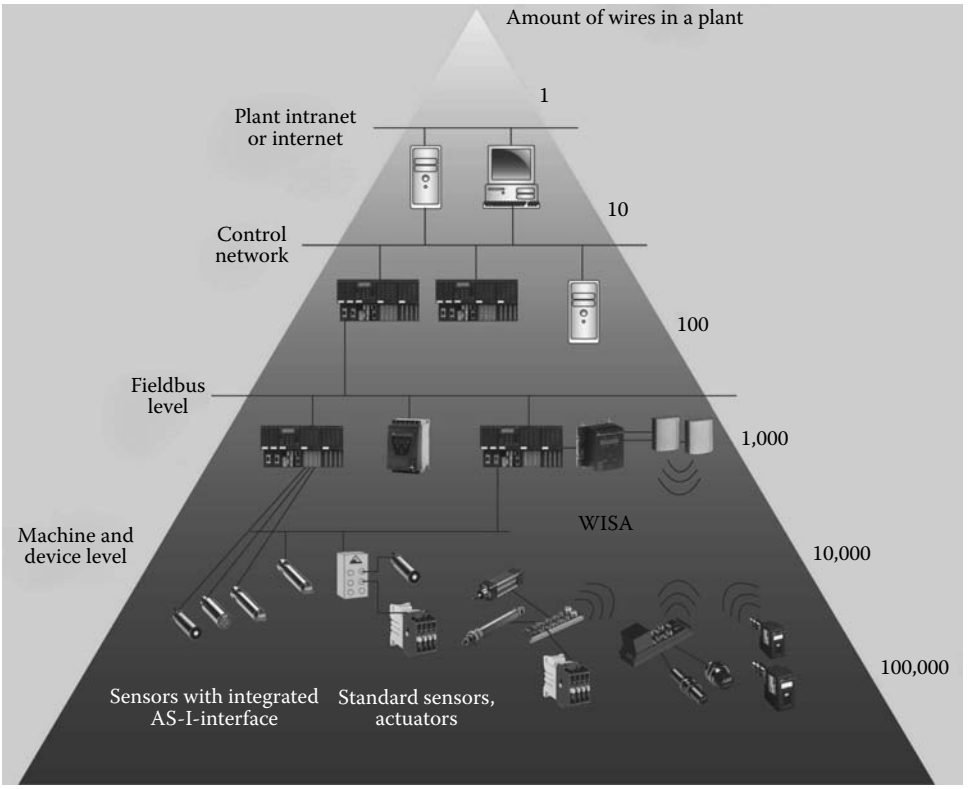


FIGURE 28.1 Automation hierarchy in factory automation, communication levels, and number of possible nodes to be connected wirelessly.

like mobile operators, controller to controller communication, and the machine and device level. It is obvious that especially in the machine- and device level a large number of wires and connectors are operated in a harsh environment.

Sensors and actuators in the machine and device level operating over wireless links:

- Allow a very flexible installation, a significant advantage in many parts of the machine-building industry, where often a custom or special purpose machine is built just once. The installation and commissioning often starts already in parallel to the ongoing design process.
- Enable fully mobile operation of sensors and actuators.
- Enable new sensor or actuator applications which would not be possible with wired devices.
- Avoid cable "wear and tear" problems.
- Avoid costly and time-consuming integration of mechanical cable tracks/chains, slip-rings, or similar mechanical elements.

In an automotive assembly plant, up to 100,000 input or output points (IO) may be wired and many of the cables and connectors are subject to repeated movement or reconnection.

Wireless communication systems for the machine and device level in factory automation must guarantee high reliability, and low and predictable delay of data transfer. The requirements in the

machine level are mission-critical and considerably more stringent than typical requirements for other areas and especially compared to home, building, or office applications.

Requirements on power supply and real-time communication for such applications are highly demanding, and cannot be satisfied with today's off-the-shelf wireless systems.

Wireless systems offer advantages in terms of availability, cost, and flexibility also in the device and machine level of factory automation. To gain most of these advantages a truly wireless solution, where communication and power supply are wireless, is needed.

A novel factory communication system, called wireless interface for sensors and actuators (WISA), has thus been developed which provides both wireless communication and wireless power supply. The power supply is based on magnetic coupling while the real-time wireless communication uses a collision-free time division multiple access (TDMA) protocol, combined with frequency hopping (FH) and is based on the physical layer of IEEE 802.15.1. WISA provides reliable and low delay transmission for a high number of nodes.

This chapter explains the practical requirements and the system specifications, the design concepts for communication, and power supply and describes practical experience gained especially with respect to reliability and coexistence with other wireless systems. It further derives simple performance profiles for WISA and other wireless technologies being discussed for industrial use, based on the requirements in factory automation.

28.2 WISA Requirements and System Specifications

The basis for system specification of WISA has been a thorough analysis of the requirements of typical applications of the targeted machine and device level:

- **Roundtable production machines**: These are used in various forms in many automated discrete production facilities. They can have up to 10 devices per m^3 machine volume (300 per machine). Many of these devices have a relative movement with considerable stress on wires and connectors. Most of them are also triggered at different times in one production cycle. Cycle times may vary from 1 s (one product produced and tested per second) in many steps to 1 min.
- **Production lines**: An automated line-type machine assembling, testing, and packing a standard consumer or household product may have 2000 (1000–3000) IO points.
- **Automotive "robot production cells"**: They have an IO point density of around two devices per m^3 machine volume (200 per $7 \times 6 \times 2\,m^3$). Sometimes, several cells are collocated and work together.

All of the above applications are typically found several times in a factory hall (5–20) often closely together.

In an "average automotive assembly plant" ($150 \times 150 \times 2\,m^3$ machine/application space), around 100,000 IO points can be encountered, leading to an average wireless device density requirement of around $2/m^3$. Assuming that only some of the devices are wireless and some are wireless IO concentrators with, e.g., 8 IO points, an average figure of a suited technology should be at least 0.25 wireless devices/m^3.

There are two related basic wireless requirements in such applications:

- Low additional latency due to the wireless link (e.g., < 10 ms) to
  - Not unduly increase the cycle time of the application, which otherwise would mean lost product output. Fifty devices (or IO points) triggered independently twice and

always having to wait 10 ms for each signal to arrive would mean 1 s of additional cycle time already (a 100% increase in a fast roundtable application!).

- Stay within time limits given by the application (e.g., a critical mechanically controlled movement) which otherwise could mean destruction of the machine, tools in the machine, or at least the produced product.

- Very high reliability to get a signal within a defined upper time limit (e.g., to avoid the above described destruction).

In contrast to other wireless use cases like mobile operators and service, the very high reliability to get a signal within a defined time limit is an absolute requirement.

Practical values for the error rate of not getting the signal in time depend on the industrial use case assumed. A typical round table application can have easily 300 IO points, mainly sensors, of which 100 are here assumed to be wirelessly transmitted in the following Table 28.1. A round table is a typical arrangement in factory, where a stepwise rotating table is used to do many different assemblies, testing or loading/unloading steps in parallel. The substations on the table have additional relative movements, so wireless is an issue due to 2D movements. Typically most of the IO signals are digital and the events are typically separated in time (unsynchronized), resulting in many telegrams, worst case equaling two times the amount of IO per production cycle (see Table 28.1).

In principal, the distribution of wireless latencies will qualitatively look like in Figure 28.2 in practical implementations. Sketched is a coexistence situation with other frequency users. If the wireless

TABLE 28.1 Events in a Roundtable Production

	1 Production Step per 2 s Event per Second
100 wireless nodes or IO points[a]	≥100 telegrams per second
Per hour	≥360, 000 telegrams
Per day (3 shifts)	≥8 mio. telegrams
Per year (w/o weekends)	≥2, 400 mio. telegrams

Note: Less than one failure can be tolerated per year, resulting in a needed packet error rate of $\ll 10^{-9}$.

[a] Digital IO assumed, concentrating IO nodes would have the same amount of telegrams, as only rarely IO points are simultaneously triggered.

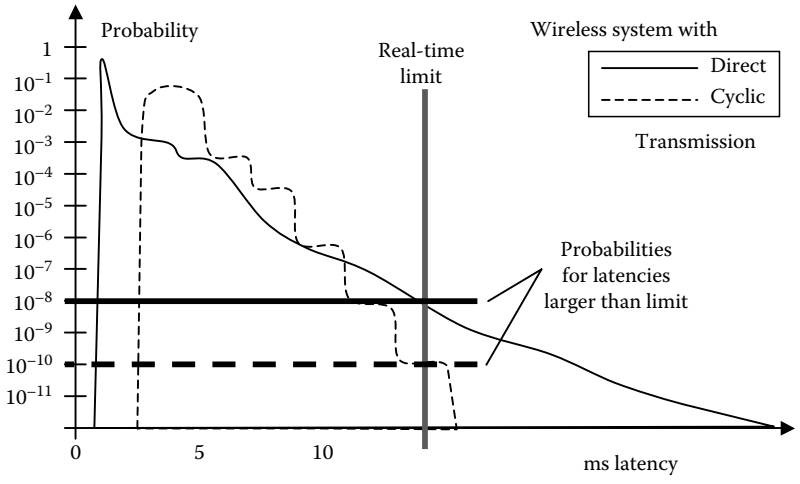


FIGURE 28.2 Typical distribution of latency measurements in wireless automation devices.

system has a cyclical transmission, e.g., as in WISA, a distribution pattern with steps can be noted (---- curve).

In practical applications, a large difference between the two curves can be noted by comparing the error rate for keeping a certain time limit, like sketched with the vertical bar. A factor of 100 results which means in the round table application that with one wireless system (---- WISA) there is an error, e.g., a production stop or even more severe problem, to be expected less then once a year, in the other case nearly once a week. A dramatic difference was seen from the user side.

The wireless communication subsystem transmits messages between the sensors/actuators (S/A) and a base station. The communication requirements derived from the above include

- Fast response times (generally less than 15 ms).
- (Practically) guaranteed upper time limit for the response time (e.g., 15 ms on the wireless interface with a miss probability of less than 10^{-9} under normal operating conditions).
- Guarantee a high reliability of data transmission even in the unfriendly environment of factories, where radio propagation may be affected by many obstacles and where various sources of interfering signals must be expected, e.g., with other users of the same frequency band.
- Have low power consumption, which is crucial due to the limited capacity of any wireless power supply.
- Serve a large number (up to 120) of sensors/actuators to one base station.
- Permit coexistence of multiple cells: Up to 300 devices in a single machine have to be supported.
- Have a practically not limited number of sensors/actuators in a large area (>1 device per m^3 volume to cover 100,000 nodes in a factory).
- Permit coexistence with other wireless systems.
  - Be robust.
  - Efficient use of the valuable frequency band resource.

Currently, the most suitable radio frequency band available for high-bandwidth, short-range wireless communication is the industrial, scientific, and medical (ISM) frequency band at 2.4 GHz. Radio transceivers operating in this band are readily available, e.g., radios adhering to the IEEE 802.15.1 standard (physical layer as used in Bluetooth).

The ISM Band also has a good balance between possible disturbances (more likely at lower frequencies, see Figure 28.9) and a good range. Effective range is lower at higher frequencies with a given transmit power. Electromagnetic interference is a concern, however measurements in different environments (see Section 28.4.3) have shown that typical industrial electromagnetic noise is significantly reduced above 1 GHz, with only arc welding providing broad band noise up to 1.5 GHz.

The protocols of existing standardized wireless systems designed for other purposes do not satisfy the requirements of the sensor and actuator level in machines used in automated factories:

- Wireless local area networks (WLANs) (IEEE 802.11) [12] and Bluetooth (IEEE 802.15.1) [11,13] and also forthcoming systems [15], including ultrawideband (UWB) systems, are all designed for high throughput between a small number of terminals, with less stringent latency, reliability, and power requirements.
- ZigBee (IEEE 802.15.4 based) [14] cannot reliably serve a high number of nodes within the specified cycle time.

- Specialized low energy protocols from building automation like Enocean [15] are not reliable and real-time suited as defined above.
- Passive electronic tagging systems (radio frequency identification tags [RFID]), as used for electronic article surveillance, do not have sufficient range, speed, and flexibility.

Hence, it was necessary to design a new communication protocol for WISA, which, however, uses much of the available standard hardware.

28.3 Communication Subsystem Design

The radio characteristics of the WISA system are those of the IEEE 802.15.1 physical layer specification used in Bluetooth [4]. The raw data rate is 1 Mb/s and transmit power is 1 mW. This comparatively low transmit power is appropriate and sufficient for a range 5–10 m, as required for the typical node and base station density.

Not so obvious is that the low transmit power and therefore limited range is also dictated by the requirement of a sufficiently large average node density. A good comparison is human communication, where also a large number of people, e.g., at a cocktail party can communicate reliably and fast, due to fairly low voice/range, which allows others to reuse the transmission medium in a certain short distance. Just a few participants increasing their transmit power/range will disturb many others.

28.3.1 Medium Access and Retransmission

TDMA (one fixed time slot per device) with frequency division duplex (FDD) (device and base station can send at the same time) and FH is used, as illustrated in Figure 28.3. The downlink transmission (from the base station) is always active, for the purpose of establishing frame and slot synchronization of the S/As. Uplink transmissions from an S/A occur only when it has new data or an acknowledgment to send (event triggered). WISA allows up to 120 S/As per cell in different time/frequency slots.

The TDMA frame length, T_{frame}, is 2048 µs. Each S/A may transmit in one out of 30 slots per frame. To support up to 120 S/As per cell, each S/A is part of one of four uplink groups $\{\text{UL}_1, \text{UL}_2, \text{UL}_3, \text{UL}_4\}$ with separate frequencies. The five frequencies hop synchronously at each frame boundary.

The slot structure is shown in Figure 28.4. The downlink slot format is a compromise between the efficiency of channel usage and synchronization latency of the downlink.

A device transmits its data packet in its assigned uplink time slot. An acknowledgment is expected in the corresponding downlink slot, which is staggered by 256 µs to allow for TX/RX-turnaround time (see Figure 28.3). If no acknowledgment is received, the S/A will retransmit the data packet in the next frame. With frame-by-frame FH, the radio channel used for retransmission will largely be independent of the previous transmission, thus increasing the probability of successful transmission.

28.3.2 Frequency Hopping

Figure 28.5 shows the measured frequency-depending attenuation in the 2.4 GHz ISM band. It can be seen that the coherence bandwidth of the signal may be some tens of MHz (corresponding to propagation path length differences of 30 m). Wideband FH will thus improve transmission performance. It also reduces the effect of other devices that operate in this band. The global system for mobile communication (GSM) [5] and Bluetooth [4] systems are well-known wireless communication standards using TDMA and FH. Their FH sequences are determined in a pseudorandom manner and therefore do not control frequency separation between consecutive hops. Hence, consecutive transmissions may have similar high packet error probability. It is for WISA of interest to employ FH sequences

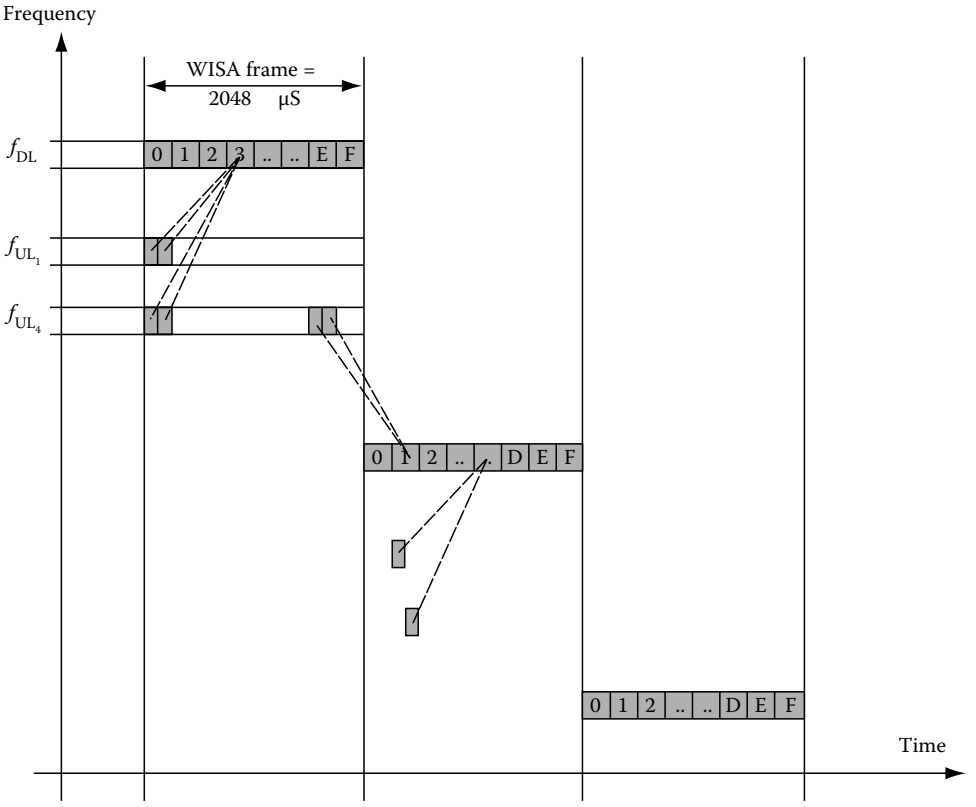


FIGURE 28.3 WISA TDMA/FDD/FH pattern. Dotted lines connect downlink/uplink slots allocated to one sensor/actuator. (TX/RX-turnaround time = 2 double slots = 256 µs, see Figure 28.4). (From Scheible, G., Dzung, D., Endresen, J., and Frey, J.-E., *IEEE Ind. Electron. Mag.*, 1, 25, 2007. With permission.)

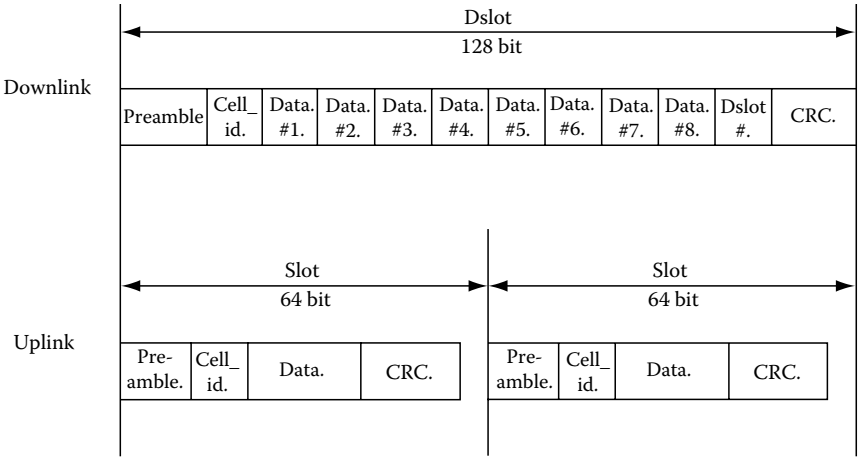


FIGURE 28.4 WISA slot format: downlink double slot duration = 128 µs, uplink slot duration = 64 µs. Double slots may be used on the uplink for higher payload data profiles. (From Scheible, G., Dzung, D., Endresen, J., and Frey, J.-E., *IEEE Ind. Electron. Mag.*, 1, 25, 2007. With permission.)

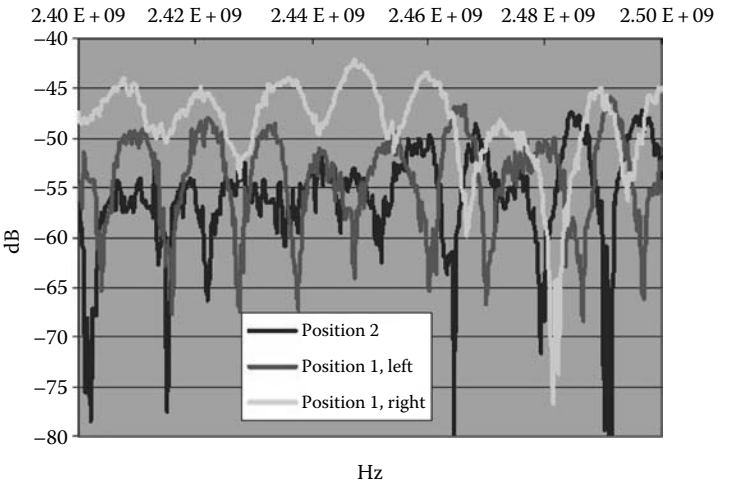


FIGURE 28.5 Frequency selective fading in the 2.4 GHz ISM band: frequency-dependent attenuation from a sensor/actuator to three BS antennas in a test installation. Gray and light gray: pair of base station antennas, separated by 37 cm. Black: third BS antenna, line of sight path obstructed by a robot arm. (From Schieble, G., Dzung, D., Endresen, J., and Frey, J.-E., *IEEE Ind. Electron. Mag.*, 1, 25, 2007. With permission.)

with the property that consecutive hops are widely separated in frequency. The WISA use of the ISM band of 80 MHz is partitioned in seven disjoint sub-bands, each containing 11 hop frequencies at a given channel spacing of 1 MHz. The sub-band bandwidth of 11 MHz is normally larger than the typical bandwidth of the fading, and the seven sub-bands together cover the available band. The method produces 60 different FH sequences with low cross-correlation. The hopping sequences have a period of $7 \times 11 = 77$ frames. Each hopping frequency is visited exactly once per period. An example of the five FH sequences for a given cell_id is shown in Figure 28.6.

In summary, the constructed FH sequences have a number of desirable properties:

- Consecutive frames are in different sub-bands.
- (Duplex-) spacing between downlink and uplink frequencies is three sub-bands apart.
- Four concurrent uplinks are in the same sub-band and hop with a fixed spacing of 3 MHz apart.

28.3.3 Antenna Diversity and Switching

To additionally improve the robustness, multiple antennas with separation of at least half a wavelength λ may be used to reduce the effects of multipath fading and shadowing. This effect is exploited by antenna diversity reception. With $\lambda = 12.5$ cm, only the base station (BS), for practical reasons, may be fitted with multiple antennas. The simplest way is to switch transmit and receive antennas once per frame in a round robin sequence.

28.3.4 Multicell Operation

The WISA system is designed such that each cell uses the entire 2.4 GHz ISM band of 80 MHz, without any need for intercell coordination, except that a different cell_id has to be set in each cell. Where cells operate in close proximity, or are even colocated, the interference depends on the correlation properties of the FH sequences and the cell traffic load. Cross-correlation between periodic integer

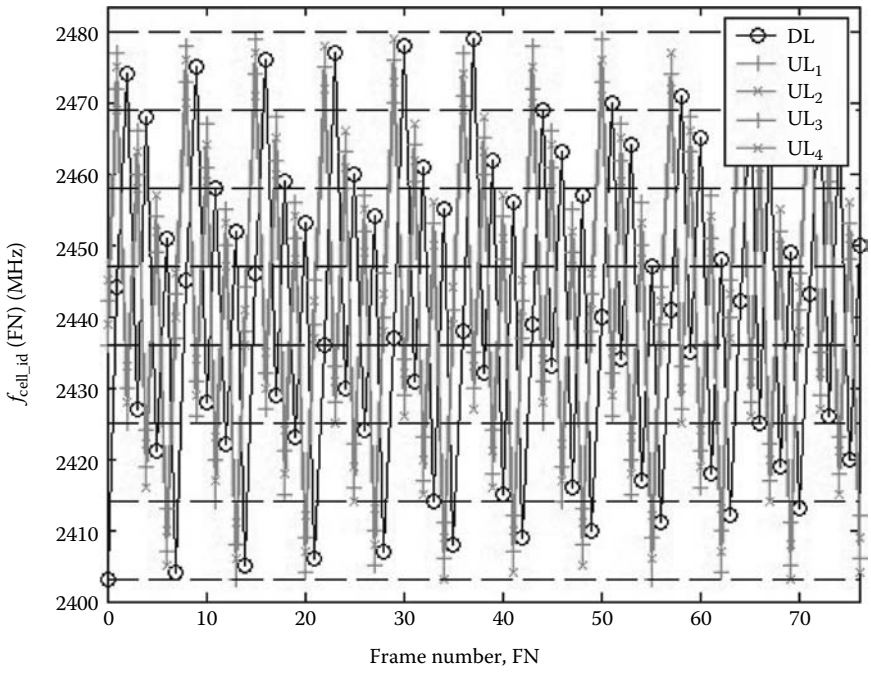


FIGURE 28.6 Example FH sequences (cell_id = 27). (From Schieble, G., Dzung, D., Endresen, J., and Frey, J.-E., *IEEE Ind. Electron. Mag*., 1, 25, 2007. With permission.)

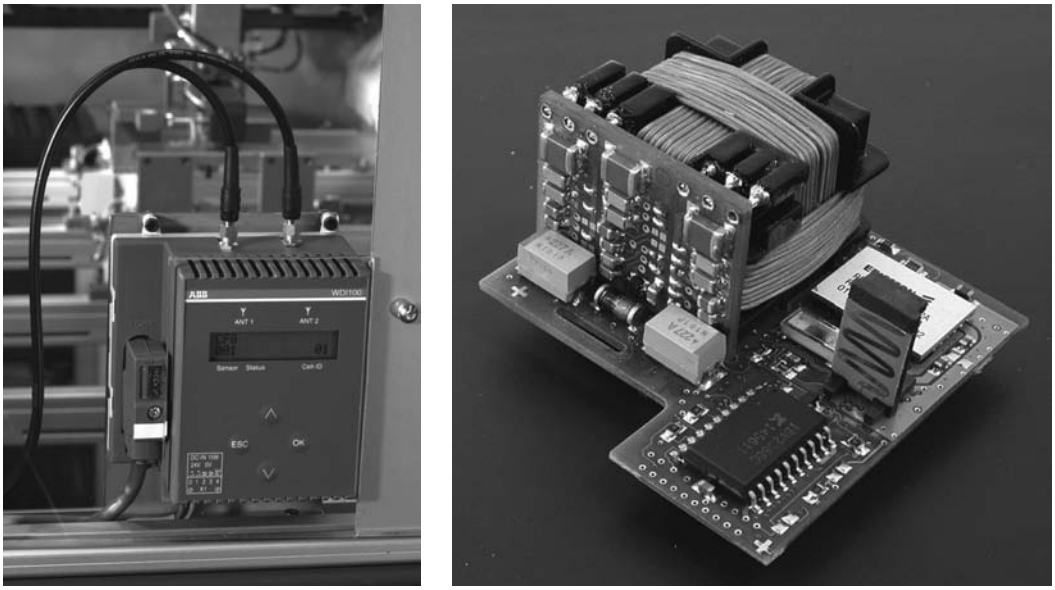
sequences has been studied in Ref. [6]. In general, there are a number of interference effects to be considered:

- Co-channel interference from same frequency transmissions becomes relevant if the C/I ratio is below 14 dB [4] (distance effects).
- Adjacent channel interference is relevant if the adjacent channel interference C/I ratio is below −20 dB. This condition may well be true for colocated base stations.
- Multiple interference from any of the five links.
- (DL, UL_1, UL_2, UL_3, UL_4) of an interfering cell within range. While downlink transmissions are continuously active, the uplink load is typically far lower. With a frame rate of 488 frames/s, and assuming a maximum uplink rate of 5 transmissions per second per S/A, a slot is only used with a probability of 1.02%.
- Multiple interference from the links of more than one adjacent cell. Links from different cells are mutually asynchronous, causing interference to occur at different times.

To assess these effects, a generalized cross-correlation was defined and its properties were analyzed. The results confirm that even in the worst case, the retransmission protocol ensures that messages can reliably be transmitted.

28.4 Communication Subsystem Implementation

Figure 28.7 shows a WISA base station, which is controlled by a microcontroller. Time-critical control of the radio frequency (RF) transceiver and baseband signal processing are delegated to an field programmable gate array (FPGA). The base station maps the wireless links to the addresses of



(a) (b)

FIGURE 28.7 WISA implementation: (a) WISA base station with antenna connectors and field bus plug (FBP) connection to controller. (b) WISA sensor module with receiver coil for wireless power supply and communication transceiver, controller, and antenna. (From Scheible, G., Dzung, D., Endresen, J., and Frey, J.-E., *IEEE Ind. Electron. Mag.*, 1, 25, 2007. With permission.)

the fieldbus connecting it to the automation controller. Diagnostics and configuration features are supported.

Due to the communication protocol requirements, a standard RF front-end cannot be used for the base station implementation. The requirements differ significantly from those found in WLANs and Bluetooth in that the BS must operate in full duplex mode. The BS must simultaneously receive signals from up to four SAs with a dynamic range of more than 60 dB.

In contrast to the BS, the WISA node may use existing RF transceiver hardware (IEEE 802.15.1 based). A WISA sensor module is shown in Figure 28.7 (in this implementation a proximity switch [2]), with the power receiving coil, an IEEE 802.15.1 based transceiver, and a microcontroller and an FPGA for the WISA protocol. For low power consumption, the transceivers "sleep" until an event occurs. The communication antenna on the module has an omnidirectional radiation characteristic in order not to have to line up the device at installation.

28.4.1 Coexistence

Wireless coexistence is a practical and increasing concern in industrial automation applications. Many different applications may use wireless systems, which may compete for the same frequency band with different technologies. The limited resource are the radio frequencies, and for WISA that is in the ISM Band from 2.4–2.483 GHz.

The situation may be compared with highways, which have a limited width and many different types of vehicles fulfilling different transport tasks. For each of the transport tasks, different requirements apply; the capabilities of the vehicles are quite different also. In addition, certain rules apply to avoid dangerous situations (e.g., speed limits).

In today's factories coexistence or frequency management has to be established to coordinate the use, as the frequency bands are a valuable resource. In a larger factory, generally every system

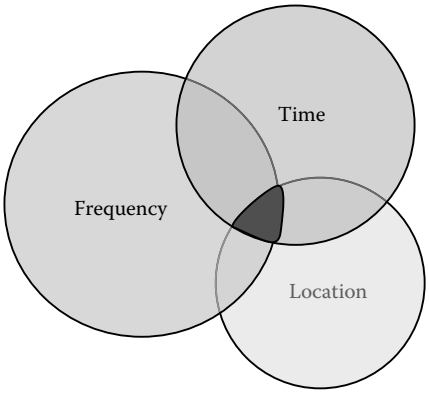


FIGURE 28.8 Coexistence issue occurs if there is a "hit" in time and frequency and location.

has to be announced in advance and released by a person/committee responsible for frequency use allowance [17,18].

Wireless coexistence is an issue only if the same frequency is used at the same time and close by (Figure 28.8).

Separation in location is seldom a free parameter; therefore, time and frequency can be optimized.

If there would be an unlimited bandwidth available, simple frequency separation would be the simplest approach. Nevertheless, the highway example shows that reserving a full lane of the highway just for one vehicle type is generally not efficient. Many different vehicles may use the same lanes if they have the ability to change lane and if on average a low duty cycle or low density of vehicles can be guaranteed.

Separation in time, by low duty cycle, is exactly the approach being used in many wireless installations by frequency management, thus allowing many users with low duty cycle.

In practical applications, coexistence always has two aspects:

1. Be robust against interference from other wireless users (low susceptibility).
2. Efficient usage of radio frequencies (spectral efficiency and low duty cycle) to leave enough free resources for other users.

WISA was designed to be robust by using adaptive retransmissions. Nevertheless, the continuous use of a significant part of the frequency band by an unfriendly system with a high duty cycle and long telegram lengths (longer than the maximum number of WISA retransmits) will make it impossible for a WISA system to transmit its data within the time limit requirement (e.g., 15 ms).

To discuss coexistence more quantitatively, characteristic numbers are helpful, which are defined here as follows:

Robustness can be measured by the factor between the maximum to the minimum delay values, measured in some defined coexistence test-setup.

Robustness **R** = min. delay/max. delay

$$R_{WISA} = 4\,ms/16\,ms = 0.25$$

In an existing system between two user interfaces under idealistic and realistic but more worst case conditions, e.g., the coexistence with other wireless systems in their typical channel occupation mode, the more close to unity, the better is the robustness.

Spectral efficiency can be measured as the ratio of data rate divided by the used bandwidth.

Spectral Efficiency $\mathbf{E} = D/B$ (D, data rate in b/s; B, bandwidth in Hz)

$E_{\mathrm{WISA}} = 1\,\mathrm{Mbit/s}/1\,\mathrm{MHz}$

This is an implementation and application independent value which enables technology comparison.

A more practical performance figure when looking at implemented solutions like WISA would be

Performance $\boldsymbol{P} = 1/(T_s/n_U * B)$

(T_s, slot time in ms);

n_U, number of parallel uplinks)

= 4 for WISA

= 1 for symmetrical systems

$P_{\mathrm{WISA}} = 1/(0.064\,\mathrm{ms}/4 * 5\,\mathrm{MHz})$

= 12.5 nodes/ms/MHz

The frequency usage can give an indication of the interference potential of a wanted system to other wireless systems. A simple practical value can be derived from multiplying the time/duty cycle and the bandwidth used

Frequency usage $\mathbf{F} = T * B$ (T, fraction of time; B, bandwidth)

$F_{\mathrm{WISA}} \sim 1 * 1\,\mathrm{MHz}$

This gives a very practical figure of how much of the available frequency band, here, e.g., the ISM band as a very valuable resource is used for a certain implementation. The frequency usage should of course always be as small as possible; nevertheless, the resulting interference depends in practice mainly on the type of the other/victim system. Different victim systems react very different to frequency usage by a wanted system, dependent on if they had been designed for coexistence or not. WISA, e.g., uses in average not much more than 1 MHz of bandwidth continuously, e.g., in an application as defined in Table 28.1.

Another practical aspect of coexistence is stability. The influence of one system on another should be as predictable as possible for automation systems. WISA has in practical applications a nearly constant frequency usage, which is determined mainly by its downlink, thus making its effect on other systems relatively predictable. The short uplink telegrams generate on average only low traffic load (<10%), even with high number of nodes and events.

Other wireless systems in addition to WISA used in industrial applications are

- IEEE 802.11, WLAN: In factory automation, WLAN is applied in many plants for varying applications. There are two typical uses:
  - Access of many different applications to the plant/IT network (scanners, order data transfer, production data transfer, network connection with, e.g., a notebook).
  - Communication to AGV systems: Automated guided vehicles which fulfill transport needs.

WLAN has its strength as a widespread standard allowing a large number of different clients to access the same network. Typically delay times below 100 ms are not an issue here. WLAN is very susceptible to disturbance as it has a large receiver bandwidth and its carrier sense multiple access (CSMA) and

clear channel assessment (CCA) features—designed to enable other WLAN devices to access the same system—also reacts to narrow band frequency user inside the used band. Therefore, the latency of a WLAN system will have a large jitter in practical applications. Measurements have shown values to vary from 1 ms to sometimes over 30 ms in a coexistence situation with other low duty cycle users.

- Bluetooth: Bluetooth is implemented in many mobile phones and notebooks; therefore, it is likely to be present. Bluetooth was made for user device interaction and is a robust protocol. It is suited for HMI purposes in industrial applications. Bluetooth devices do not use CSMA and may thus have an effect on WLAN links (especially in the search/pairing mode).
- IEEE 802.15.4/Zigbee/WirelessHART: Such devices will be used to cover a larger area for condition monitoring or building automation use cases. They will typically have a low duty cycle and are not to be expected in large quantities in factory automation (have, therefore, not been considered as disturber for WISA in more detail). Such devices might be disturbed by a WLAN with a high duty cycle if in close distance and with not enough frequency separation (except Wireless Hart with an FH-option).

To estimate distance-dependent interference effects, an empirical model for the path loss (PL) is often used for Bluetooth-type applications (see, e.g., [7]). In the near field of the transmitting antenna, this attenuation is proportional to d^2, where d is the distance (6 dB additional PL if distance is doubled). For larger distances the empirical formula stipulates attenuation proportional to $d^{3.3}$ (10 dB additional PL if distance is doubled). This is a coarse model; hence, the distance values derived below should be considered only as indicative.

28.4.1.1 Interference from IEEE802.11b

Consider factory staff working in the vicinity of a WISA cell using a laptop computer to upload a large document from an 802.11b base station. The WISA transmission power is 0 dBm. Using the PL model above, this results in a carrier reception level C, at the WISA base station at a distance of 5 m, of $C = -54$ dBm. IEEE 802.15.1 receivers require a co-channel carrier-to-interference ratio of $C/I >$ 14 dB for adequate reception performance. Hence, the interferer level should be $I < -54$ dBm-14 dB $=$ -68 dBm. The 802.11b transmitter power may be up to 100 mW in 22 MHz bandwidth, equivalent to 6.5 dBm per WISA/IEEE802.15.1 receiver bandwidth of 1 MHz. The required PL to ensure safe operation is therefore 6.5 dBm $-$ $(-68$ dBm$)$ $=$ 74.5 dB. This translates into a minimum safe distance requirement of 25 m for an 802.11b transmitter.

28.4.1.2 Interference from Bluetooth

Assume a factory worker using a Bluetooth phone headset. The typical Bluetooth transmitter power is 0 dBm at 1 MHz bandwidth and the required PL to ensure $C/I > 14$ dB is therefore 68 dB, which corresponds to a minimum distance requirement of 16 m.

28.4.2 Performance Measurements

WISA operation is at relatively short distances, with reception signal levels of about -60 dBm, well above the sensitivity level. Hence, performance is in practice not limited by distance or fading effects, but mainly by robustness to external radio interference.

Performance evaluations were done by measuring the message delays. Any loss of message in the uplink (sensor status message) or in the downlink (BS acknowledgment) causes a retransmission in the next frame, resulting in an additional delay of 2048 μs. A measurement setup consisted of 60 WISA nodes set to transmit at a rate of two messages per second over distances of about 70 cm.

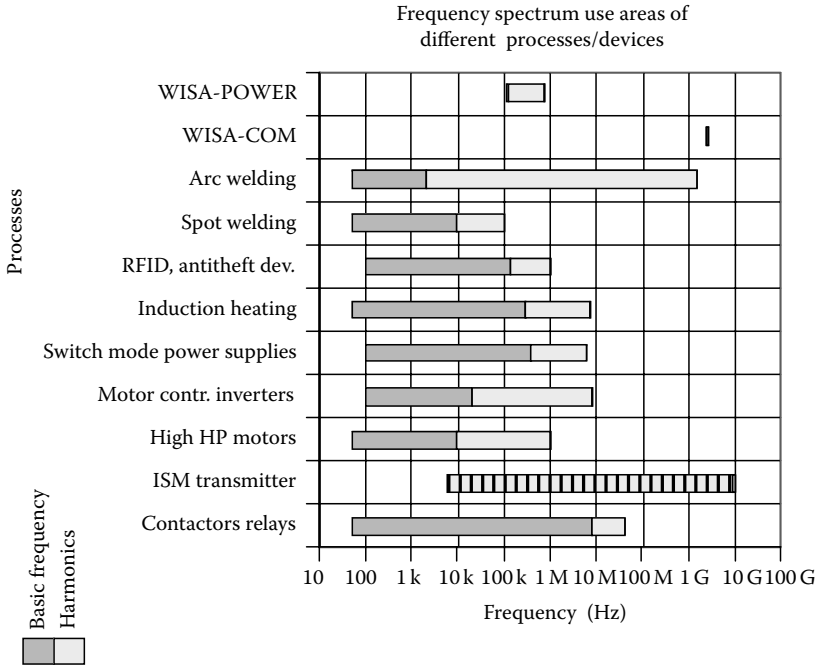


FIGURE 28.9 Electromagnetic interference from typical applications in industry.

In the interference-free case, one in about 1000 uplink messages required a retransmission (marked as "reference" in Figure 28.11).

The effect on the WISA transmissions due to interference from other WISA base stations was also measured in a realistic scenario. The antennas of the interfering base stations are set to point toward the WISA system under test, thus simulating multicell operation as shown in Figure 28.11 (top).

28.4.3 Industrial Electromagnetic Interference

Investigation of a number of potential industrial sources of interference showed that no interference is to be expected in the 2.4 GHz band from high power welding equipment or frequency converters. Figure 28.9 gives an overview of typical industrial devices, their basic frequency, and indications for their harmonics in the spectrum (Figure 28.10).

The only interference may come from legal ISM transmitters. They are not widespread but have to be taken into account if unshielded. They can have very high power levels (e.g., microwave drying applications in the kW range).

A spot welding gun used in induction welding, e.g., operates at up to 20 kA and generates very high electromagnetic field strengths. However, the majority of the high frequency noise decays strongly above 1 GHz and therefore has a minimal impact on WISA. The only measurable interference with high frequencies (up to 1800 MHz) was identified as being generated by arc welding. WISA devices have been tested with positions only a few centimeters away from the welding spots.

28.4.4 Interference from Other Communication Systems

A number of measurements and tests have been conducted to quantify the effect of interference from other communication systems in the 2.4 GHz band. In a first scenario, interference was caused

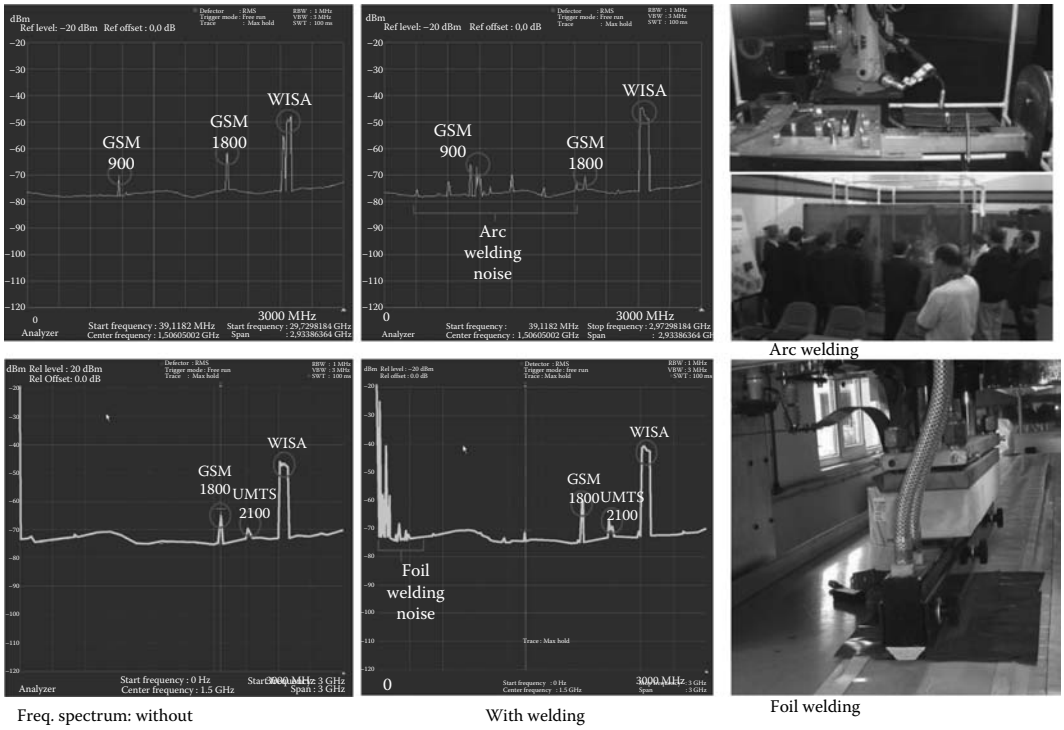


FIGURE 28.10 Frequency spectrum plots of electromagnetic noise of industrial welding equipment.

by a laptop PC with an IEEE 802.11 g WLAN communicating to an access point (AP) transmitting at 54 Mb/s on channel 5 at 100 mW. Measurements were repeated at several distances (d = 7.5, 4, and 1 m) between the AP and the WISA BS, and were taken over 100,000 WISA messages each. Figure 28.11 (middle) shows the increased number of WISA retransmissions due to the WLAN interference.

The effective data rate of the WLAN was reduced from 3.1 MB/s in the interference-free case to 2.8 MB/s (d = 7.5 m), 2.25 MB/s (d = 4 m), and 0.86 MB/s (d = 1 m) respectively.

In the second scenario, a Bluetooth-equipped laptop continuously transmits toward a Bluetooth AP at 1 mW and 1 m distance (Figure 28.11, bottom) shows that the effect of Bluetooth interference on WISA retransmissions is much smaller than the effect of WLAN interference. In the converse direction, the Bluetooth data rate was reduced from 29.2 to 25.6 kB/s by the interference from WISA.

Further work on WISA is under way on implementing WLAN options, which allow either for the downlink or all links a blacklisting of WLAN channels. As the dominant influence on WLAN is the continuous downlink (compare Figure 28.12 top and middle), just blacklisting the downlink is improving the WLAN statistic significantly, see Figure 28.12 bottom, where the downlink has been in a wire to verify uplink influence. This statistic is already very close to the reference measurement (top).

It has to be noted that generally the disturbance range of a system, outside which, e.g., frequency can be reused, is significantly larger than the safe operation range. The disturbance range of a wanted system also varies with respect to a victim system, so it is not a wanted system property only.

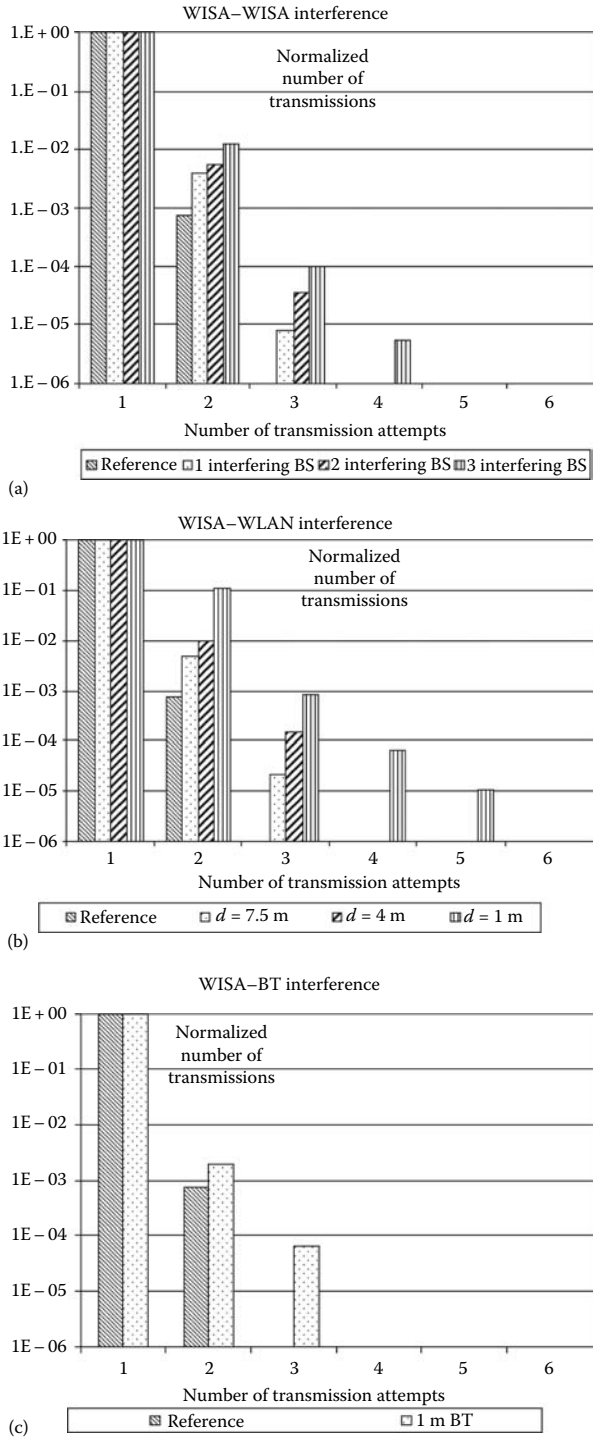


FIGURE 28.11 Measured retransmission statistics for WISA uplink messages due to interference (1 retransmit = 2048 μs). (a) WISA–WISA (1 m distance), (b) WISA–WLAN, (c) WISA–Bluetooth. (From Schieble, G., Dzung, D., Endresen, J., and Frey, J.-E., *IEEE Ind. Electron. Mag.*, 1, 25, 2007. With permission.)

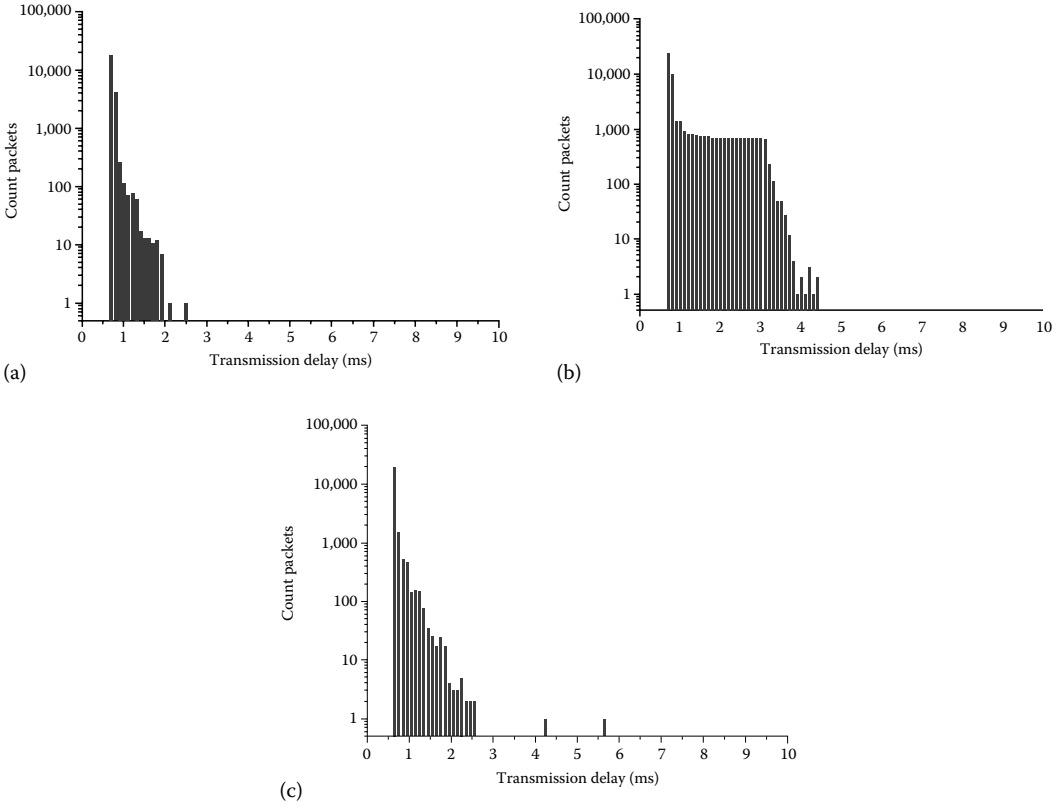


FIGURE 28.12 Measured WLAN-latency statistic (64 byte every 5 ms). (a) Reference measurement (no other frequency user). (b) with WISA in parallel, +2 ms downlink influence clearly visible (3 m distance). (c) same, but uplinks only (downlink in cable); four consecutive WISA nodes (50 events per second simultaneously)

28.5 Wireless Power Subsystem

Wireless power in principle can be supplied by different concepts:

- *Energy scavenging*: Taken from the local environment in the form of light, heat, and vibration/motion
- *Energy storage*: Included in the system in the form of batteries, fuel cells, etc.
- *Energy distribution*: Transmitted to the system via optical or radio frequencies, sound, etc.

The use of battery power is considered acceptable in the consumer world. However, in general industrial applications, where hundreds of devices require constant, reliable power supply and run day and night, batteries are not an option. Their energy density (normally around 1.2 Wh/cm^3 for primary batteries) is still too low, or the other way round the consumption of low power radios and sensors is still too high for general use. While for secondary batteries, there has been quite some development their volumetric energy density are still significantly lower than for primary batteries, typically by at least a factor of 3–5. For primary batteries, there has not been a noticeable progress regarding energy density.

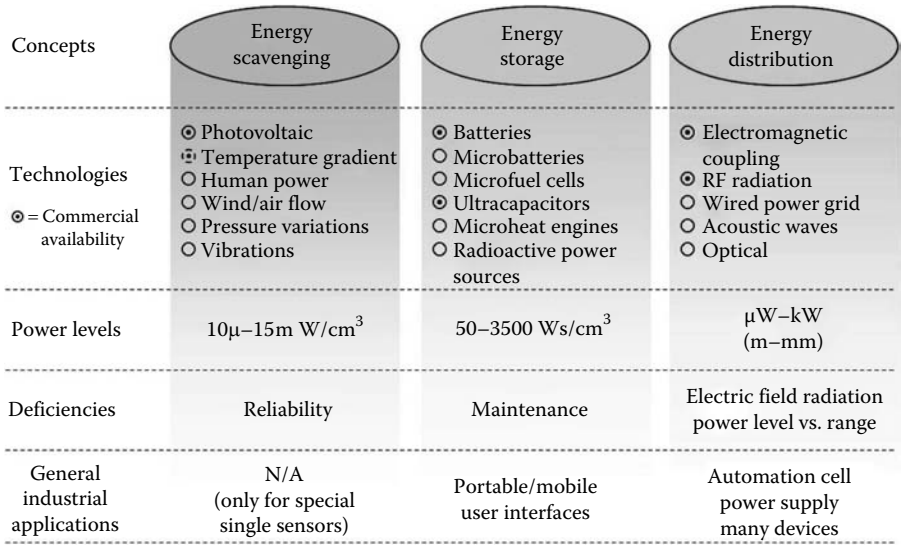


FIGURE 28.13 Overview of wireless power concepts and technologies; power levels, deficiencies, and remark for general application in factory automation.

Fuel cells are potentially somewhat better, but even their realistic potential is little more than 2 Wh/cm^3 (petrol energy density would be around 10 Wh/cm^3) and much development is still required before they could envisioned to be used in everyday industrial installations [8]. Another fact is to be considered for fuel cells. Like human beings, they simply need air to breathe and evaporate water, a challenge for IP67 housings.

Environmental energy sources also fail to meet the needs of general industry applications, due to their unpredictable nature—both in terms of general usability without special engineering work and reliability. Such solutions would also incur considerable engineering and design costs for every single device. Nevertheless, for some niche applications, there are use cases, e.g., vibration or high temperature monitoring (Figure 28.13).

Regarding the energy distribution concepts, only magnetic coupling in the near field can transfer enough power for general use. There are a number of possibilities to use magnetic coupling, depending on the transmission distance, a wide range of applications and power levels can be covered (mW at several meters of air gap up to tenth of kW at millimeters).

After a thorough evaluation of the various available options [8,9], it was concluded that the only viable, generally applicable solution for factory automation is based on long-wave radio frequencies, a form of inductive, magnetic coupling.

In Table 28.2, a comparison for a special use case of a wireless proximity switch is summarized. A figure of merit is defined, which summarizes the characteristic figures:

- Energy amount over 5 years (Wh)
- Energy cost over 5 years (USD)
- Risk of developing such a supply (factor)

The figure of merit is computed by dividing energy by cost and by risk according to Table 28.2. Inductive energy transfer at low frequencies of around 100 kHz has clearly the highest figure of merit regarding the intended application.

This is the so-called WISA-POWER approach. This wireless supply principle described below in more detail provides power supply across a distance of a few meters with the help of magnetic fields

TABLE 28.2 Example Power Supply Comparison for a Typical Autonomous Industrial Sensor Application by a Figure of Merit

Energy Source	Power	Units	Energy (Per Life of) 5 Years; Wh	Av. Power w/o Service Over Life in mW	Cost (No Service) w/o Encapsul. USD	Figure of Merit Wh per USD/Risk	Risks w/o Power 1 = Very Low; 4 = Very High	Risk Descr.	Remark
PV 1000 Lux a-Si	0.034	mW/cm^2	9.0	0.21	0.3	15.0	2	Shielding/orient	1/3 surface
PV 100 Lux a-Si[a]	0.0029	mW/cm^2	0.8	0.02	0.3	1.3	2	Shielding/orient	1/3 surface
PV 1000 Lux GaAs[a]	0.102	mW/cm^2	26.9	0.62	3	3.0	3	Rp, costs, shielding/ orient	1/3 surface, 3[a] eta a-si, 1ß[a] Cost a-Si
Thermal (thin film)[a] $dT = 10$ K	0.0063	mW/cm^2	1.0	0.02	1	0.3	3	Reliability, costs, feasibility?	Mounting area, $dT = 10$ K
HF 100 kHz (7 A/m)	1	mW/cm^3	105.7	2.41	1	52.8	2	High Q, E-smog shielding	Simple test-setup 0.5 m, 2 A, not yet optimized
HF 13 MHz	0.23	mW/cm^3	24.3	0.55	1.5	8.1	2	High Q, radiation shielding	Ps = 100 mW, antenna 3 cm, distance 1 m
HF 433 MHZ[a]	0.3	mW	13.1	0.30	2	3.3	2	High Q, regulations	Ps = 100 mW, antenna 17 cm, distance 1 m
Batterie $LiSoCl_2$	1.1	Wh/cm^3	2.7	0.06	2	1.3	1		1/3 volume
Batterie AlMn	0.32	Wh/cm^3	0.8	0.02	0.2	3.9	1		1/3 volume
Fuel cell, H_2 20 MPa	0,25	Wh/cm^3	0,6	0,01	5	0.03	4	Storage, water	1/3 volume, 50% of values of larger systems
Fuel cell, hydrid H_2	1	Wh/cm^3	2.4	0.06	5	0.16	3	Water, pressure	1/3 volume, 50% of values of larger systems
Fuel cell, methanol	2.2	Wh/cm^3	5.3	0.12	2	0.66	4	Water, technology/time	1/3 volume, 50% of values of larger systems

Note: One-third of sensor: 6 cm^2 usable surface, 2.5 cm^3 of usable volume.
[a] Estimation.

and is suitable for most sensors and electrical actuators (like pneumatic valves) in discrete factory automation.

28.5.1 "Magnetic Supply": WISA-POWER

The basic principle of a magnetic field-induced power supply can be described by the well-known transformer principle. A power supply unit feeds a primary winding, a large coil, which can be arranged around a production cell. The secondary side consists of a practically unlimited number of small receiver coils, see Figure 28.7. Each receiver coil is equipped with a ferrite core to increase the amount of flux collected by the coil and to minimize effects of external conductive materials. For this type of "transformer," magnetic coupling is very low (<0.1%). The receivable power is determined by the amplitude of the magnetic field at the location of the receiver (secondary) winding and its size.

Although human operators will rarely work continuously inside such an automated production cell, the strength of the magnetic field at all working positions (including within such a cell) complies with international occupational regulations and recommendations [10]. WISA-POWER works at the frequency of 120 kHz. The distance to be observed by a pace maker carrier to this setup would be 2.5 m at the maximum current of the WPU (24 A) according to strict German regulations

Energy losses in such a WISA-POWER system are surprisingly small and are mainly caused by skin and eddy current effects in the coil itself and in nearby metal objects, especially steel. In many different production cells in factory automation equipped with WISA-POWER, energy losses have been measured to be around 10–15 W/m^3 of supplied cell volume.

The principle does not depend on the frequency, but the chosen frequency is a good balance between

- Higher amplitudes needed at lower frequencies (higher ohmical losses)
- High frequency losses in a realistic environment, where always metal will be present to a certain degree (eddy current losses)
- Semiconductor switching losses
- Capacitive dielectric losses
- Component size (larger at lower frequencies), which is especially an issue on the receiver side, where you need to have small devices

As the receivers will typically be designed to consume just 5–100 mW (sensors–small actuators), the above-mentioned losses will clearly dominate the power consumption. Therefore, the amount of such wireless devices is practically not limited, several hundreds or even 1000 can be supplied.

28.5.2 Resonant, Medium-Frequency Power Supply

These unconventional transformers have to be operated in a "resonant" mode, to compensate for the large leakage inductances of the transformer. The resonant principle allows the wireless power unit (WPU) to stimulate the resonant circuit at relatively low voltages (<60 V).

The WPU has to have a control circuit that must also be able to accommodate:

- Changes over time in the environment, e.g., caused by the movement of large, metal objects such as robots.
- Different "load" requirements, caused by differently sized and formed primary coils (inductance values), and losses, caused by factors such as eddy currents in adjacent metal objects.
- Other nearby wireless supply systems, which may couple inductively.

28.5.3 Rotating Field

Unidirectional magnetic fields can be attenuated or blocked by metal objects, which is the case in applications where the devices have to be placed arbitrarily or are moved, e.g., by a robot throughout the application. To minimize the shielding effects, two primary loops can be mounted orthogonally (see Figure 28.14, top). The loops are fed by separate power supplies, whose currents are phase-shifted by 90° with respect to each other. This creates a field vector which rotates at 120 kHz with a constant amplitude, which is sufficient for most practical applications. With a third loop also, an undirected field could be created by rotating the field vector evenly through the volume. Nevertheless, the additional benefit would have been small in the so far realized applications, but a much more complex control approach would be needed.

28.5.4 Omnidirectional Receiver Structure

For sufficient power output, the receiver side also has to be operated in resonant mode. To make the available power independent of the receiver's orientation with respect to the primary field vector, an orthogonal setup of three coils on a common core has been chosen (see Figure 28.7). The available power density for typical "worst-case" shielding conditions in real applications is in the order of 1.2 mW/cm^3. Typically, it is a factor of 2–4 higher. The absolute power level can be modified with the secondary coil size and shape (and the distance to the primary coil). With optimized Q-factors of the receiver coils, much higher levels could be transferred, nevertheless only under ideal, nonrealistic environmental conditions.

28.5.5 Example Setup

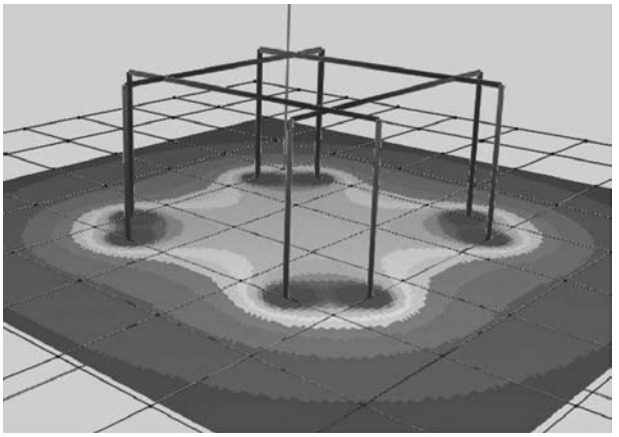
The resulting field distribution in an installation with rotating field to cover a larger volume is shown in Figure 28.14. The only design quantity needed for the loop placement is the distance d to the next parallel loop. This distance is optimal around two-third of the smallest dimension S of a loop. The smallest dimension S is defining the range of the loop antenna.

To reach the required minimum field strength for safe operation of the devices, a certain current has to be set at the WPU feeding the loops. A typical current is 24 A for loop with S = 3 m, respectively smaller in loops with a smaller dimension. If an application is larger, more loops can be cascaded. Also, a number of simpler setups are possible for special cases (line and spot type primary loops) as often the range of movement of the devices is limited. A line-type primary loop could be as long as 40 m per WPU unit.

28.6 Practical Application and Performance Comparison

The WISA technologies have led to a range of products. A fully wireless proximity switch and a sensor concentrator are using both wireless technologies and operate with special energy-optimized sensor heads. To also allow the integration of other standard sensors and typically much more power hungry actuators, which might be needed for a complete automation application, a sensor/actor distribution box with up to 16 input or outputs is available. For the WISA communication, two profiles (up to 32 bits in both directions) have so far been implemented, to accommodate the different payload data requirements of different devices (Figure 28.15).

Total power delivered even under worst case conditions to the wireless proximity switch is 6 mW (sufficient for five events per second). The larger receiver in the sensor pad WSP100, which allows the connection and supply of up to eight sensor heads, can worst case provide approximately 50 mW continuously. Both values are two to four times higher in typical working positions.



(a)

(b)

FIGURE 28.14 (a) Simulated field distribution in an example $3 \times 3 \times 3\ m^3$ setup; (b) example installation with WISA-POWER and communication showing some of the integrated power loops and the design parameter distance d, which depends on the smallest length S of the coil. (From Schieble, G., Dzung, D., Endresen, J., and Frey, J.-E., *IEEE Ind. Electron. Mag.*, 1, 25, 2007. With permission.)

FIGURE 28.15 Industrial automation devices with embedded WISA-POWER and WISA-COM technologies (left side) and necessary infrastructure (right side).

Cell dimensions which can be covered with the WPU100 power supply devices are $1 \times 1 \times 1\,m^3$ to $8 \times 8 \times 3\,m^3$ respectively $1\,m^3$ to close to $200\,m^3$ in a single cell arrangement (without vertical loop conductors inside).

Integrated into industrial IP67 products, including filtering on the device side, the fieldbus timing uncertainty, and the handling delays on the base station side of a field bus master, the performance stays within the specified limit.

The base station interface normally is a fieldbus, which adds additional latency. The base stations also support over its built-in user interface a special configurable mapping mode, which can map signals, e.g., of wireless sensors directly to other WISA devices with outputs. In that mode, very simple latency measurements can be done, without being delayed by the bus interfaces.

Measurements published in Ref. [16] show for the mapping mode a delay statistics with values between 5 and 9 ms increasing in a coexistence setup with 2 WLAN networks to max. 11 ms for a few telegrams out of 1 million measured. These latency measurements were always for two acknowledged WISA telegrams indicating that WISA works with latencies of typically 4 ms (2.5–6 ms).

The machine shown in Figure 28.14 has been in continuous shift operation since 2003. In total, several WISA applications with several hundred devices have been monitored over several years. Therefore, up to date roughly 4 million device hours have been accumulated, giving a very good feedback of the achieved reliability in production. The main lessons learned were on mechanical issues (usability of housing and connector).

Applications with up to 156 independently rotating nodes in one machine (cable winding machine, Karlskrona, Sweden) have been installed in 2005 with up to four colocated base stations. All the sensors are in cable drums which continuously move in a ferrous wheel type cage arrangement with multidimensional movement. Several more of such machines are installed or planned.

A further challenging application where the real-time properties are mission critical is automotive stamping. The delays of the wireless system have to be within the mechanical limits of the press, otherwise drastic failures could occur. The wireless devices are installed in the press tools which are frequently exchanged, so also fast start-up and reconnection times are an issue here. A WISA device

TABLE 28.3 Empirical Comparison of Wireless Technologies in Discrete Factory Automation for Short Data Packet Applications (Sensors)

I	Nodes/Network	Networks per Cell[a]	Cycle Time Air Interface (ms)	Real-Time Limit[b] (ms)	Energy/Bit with Retransmits (mWs)	Figure of Merit (Nodes per ms a. mWs)
WISA	120	3	2	15	1.2	20
BT [11,13]	7	10	8.75	50	10	0.14
WLAN [12]	100[c]	3	(100)	(500)	100	(0.006)
Zigbee [14]	7	16	15	(500)	0.6	(0.37)
Enocean [15]	20	1	(10)[c]	(1000)	0.05	(0.4)

II	Nodes per Cell	Range r in Machine in m	Realizable Avg. Node Density Per m^{3d}
WISA	360	5	2.29
BT	70	30	0.012
WLAN	300	50	0.019
Zigbee	112	30	0.020
Enocean	20	15	0.014

Source: Scheible, G., Dzung, D., Endresen, J., and Frey, J.-E., *IEEE Ind. Electron. Mag.*, 1, 25, 2007. With permission.

[a] Without significant performance change.

[b] Air interface (error rate ~10–9, at max. node density, and at five events per second and device).

[c] Not defined, value assumed for comparison.

[d] Usable volume: assumed radius of range r, height 2 m.

can, after start-up, be transmitting again already after few 100 ms, including internal secondary power supply start-up procedures.

To quantify the achieved performance, another figure of merit was defined (see right column in Table 28.3/I) as number of nodes divided by real-time limit and by power consumption. The empirical quantitative comparison of the technologies mentioned in Chapter 2 was done using the figures as given in Table 28.3. Assumed was the wireless connection of simple proximity switches (1 bit), the most widespread sensor in discrete factory automation.

Data as listed in Table 28.3 are often a subject of controversy and consistent practical data cannot be found in papers. Therefore, typical estimated values have been collected and taken to enable a comparison. Where values cannot be really guaranteed due to the technology, the figures in parentheses have been used to conduct the performance comparison. In Table 28.3, "Enocean" refers to a proprietary technology known for its low power consumption, developed for building automation.

The real-time limit in Table 28.3/I is the time delay, which can be "practically" guaranteed (very small failure probability, goal here at the mentioned number of nodes in operation: 10^{-9}).

As an example, consider a Bluetooth pico-net having up to seven slaves. Ten Bluetooth networks have been reported to operate directly in parallel [11], without a significant decrease of performance. Assuming the typical range in industrial environment, e.g., 30 m in for Bluetooth (100 mW class 1 version) and assuming a height of typical applications of 2 m, a realizable average node density figure per m^3 can be calculated (Table 28.3/II). This average figure is quite small for Bluetooth due to the longer range, compared to the requirement (0.25–2 wireless devices per m^3 in factory automation applications, see Chapter 2). Bluetooth can operate with a cycle time of ~8.75 ms, but under realistic conditions at high node densities the maximum latency that can be practically guaranteed is much higher, figures of 50 ms have been reported by various groups. For Bluetooth, a power consumption of 50 mW has been estimated, assuming a certain duty cycle in use.

The resulting figure of merit (last column in Table 28.3/I) for WISA is 20 (nodes per ms and mWs), roughly a factor of hundred higher than for any other technology. This relation changes somewhat on the energy per bit scale, with assuming a 32 bit payload, but will still be around a factor of 50 higher.

Looking at Table 28.3/II, it can be clearly seen from the achievable average node density in a factory hall, that only WISA can achieve the requirements for a whole factory.

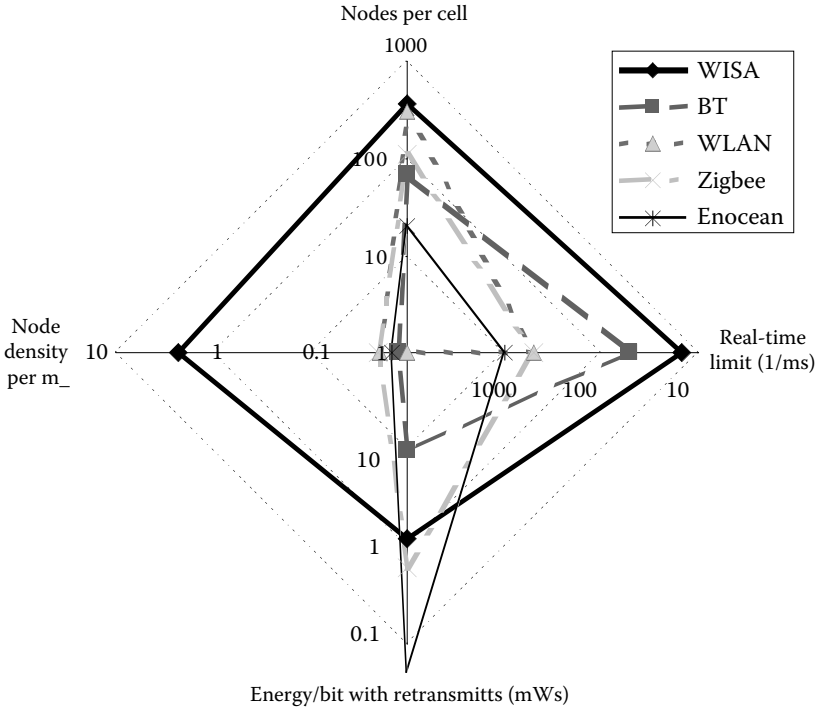


FIGURE 28.16 Visualization of comparison for the most important requirements in discrete factory automation (empirical, mind logarithmical scale). (From Scheible, G., Dzung, D., Endresen, J., and Frey, J.-E., *IEEE Ind. Electron. Mag.*, 1, 25, 2007. With permission.)

It is worth reminding that generally a smaller range allows a higher node density, due to lower interference between cells. A low range is typically acceptable in factory automation, where a given base station needs only to provide coverage for a manufacturing cell.

For technologies such as Enocean, ZigBee, and WLAN a low failure probabilities are not generally possible due to the impact of, e.g., a moving environment in machines within their long range *r* (see Table 28.3/II). Therefore the figures have been put in parentheses. Figure 28.16 visualizes the achieved match of performance of WISA compared to the requirements and the discussed competing technologies.

It should be noted that despite the empirical approach, it is not changing the picture a lot, if a certain value would be differing by a factor of two, due to the logarithmic scales chosen on each of the axes.

The characteristic figures defined in Chapter 4 coexistence can be used to help practically judge the coexistence behavior of the different in factory automation-used systems/technologies. To do this, a fictive real-time system with a TDMA approach as implemented in WISA on an IEEE802.15.1 physical layer has been assumed implemented with the different physical layers of IEEE802.15.4.-based systems (Zigbee, WirelessHART) and as comparison WLAN (IEEE802.11 g). To calculate a duty cycle for frequency usage, it has been assumed that 100 nodes should be covered as fast as possible, an application cycle time of 20 ms (e.g., programmable logic controller [PLC]) has been assumed, in which all nodes should transmit one telegram correctly.

Table 28.4 shows the assumed realizations and the values used to calculate the characteristic figures. For the 15.4 PHY, it is visible that the resulting TDMA frame of 50 ms would be much larger as the anticipated applications cycle time of 20 ms; therefore, 15.4 would not be practically usable with such

TABLE 28.4 Assumed Realizations of a TDMA Real-Time System, Based on Different Standard Physical Layers and Resulting TDMA Frame in Which All Nodes Could Transmit

Actual Nodes ms Application Cylce (PLC)	100 20 Bandwidth (MHz)	1 Byte Data Slot Time (ms)	Data Rate (Mbit/s)	Resulting TDMA Frame Time (ms)
IEEE 15.4 PHY	3	0.5	0.25	(50[a])
IEEE 15.1 PHY-WISA	5	0.064	5	1.6
IEEE 11 g PHY:	22	0.04	54	4

[a] Longer than necessary application cycle: not realizable.

TABLE 28.5 Characteristic Figures for Three Different Physical Layers in an Assumed TDMA Real-Time Implementation (Empirical, Indicative Only)

Characteristic Figures	Performance		Coexistence	
	E = Spectral efficiency	P = Performance	F = Frequency usage (at max. nodes)	R = Robustness (empirical)
	Bit Rate/ bandwidth	Nodes/cycle time and per Bandwidth	Bandwidth ∗ (nodes ∗ slot time/application cycle time)	Min. latency/max. latency
		Nodes/(ms ∗ MHz)	(MHz)	Ideal/worst case
IEEE 15.4 PHY	0.083	0.67	7.5	~0[a]
IEEE 15.1 PHY-WISA	1.0	12.5	1.32	0.25[b]
IEEE 11.g PHY	2.45	1.14	4.4	0.02[c]

[a] The worst case latency for a 15.4 system is very large, as fadings with a large bandwidth can frequently or even statically occur, that is why normally a mesh layer should be used to find alternative routes then.
[b] WISA has been measured, e.g., in Ref. [15].
[c] A WLAN latency as, e.g., measured in Ref. [16] in a very quiet environment can be as fast as 0.7 ms, but in a realistic environment, 100 nodes connected with parallel other WLANs and users, up to 35 ms can be frequently measured.

a number of wireless nodes. Table 28.5 calculates the characteristic figures defined in Chapter 3 for the three different physical layers.

The IEEE802.15.1 physical layer-based WISA implementation has, although it uses five 15.1 frequencies simultaneously, a lower frequency usage *F* compared to the much higher data rate WLAN, which, if used like that, suffers from its overhead at the assumed small data packet sizes of 1 byte (e.g., a typical simple binary sensor).

The values of Table 28.5 are plotted to visualize again the different profiles of the technologies for short packages (Figure 28.17).

28.7 Summary

A truly wireless sensor/actuator interface with wireless power and wireless communication for real-time factory has been presented. WISA uses IEEE 802.15.1 radio transceivers, but adds an optimized TDMA protocol to support a high number (120) of SAs per base station as well as short cycle times (2048 μs).

It is important to emphasize that WISA parameters differ significantly from the design goals of existing short-range wireless office systems such as Bluetooth or ZigBee. This difference has been verified graphically and by a figure of merit, which compares the performance to the most important requirements in discrete factory automation applications.

The WISA FH sequences are constructed to guarantee adequate frequency separation between consecutive hops and low correlation (low interference) in the case of multicell operation. In combination with an appropriate ARQ scheme, this provides reliable and low delay data transmission. All cells use the entire available ISM frequency band of 80 MHz at 2.4 GHz, without any requirement for frequency planning.

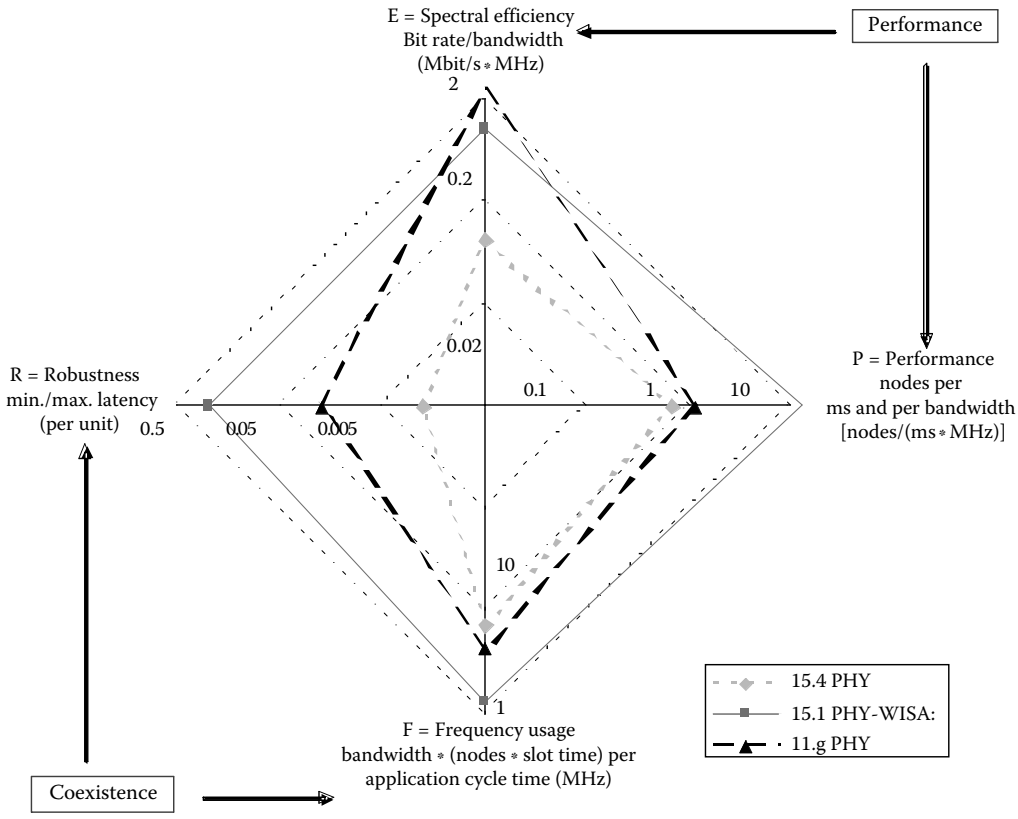


FIGURE 28.17 Characteristic figures for coexistence and performance according to Table 28.5.

The effect of interference from other systems likely to operate in the same frequency band has been analyzed, and WISA is designed to withstand such interference. For comparison of coexistence and robustness, new characteristic figures have been proposed to quantify discussions.

A wireless power supply system has been designed and verified together with the wireless devices in many practical installations.

The system has been implemented into products and its performance in real factory applications has been verified and more than 8 million node hours have been accumulated. Products based on it are commercially available and are used in a growing number of diverse applications in discrete factory automation [3]; some of them are already running for more than 5 years.

References

1. A. Willig, K. Matheus, and A. Wolisz, Wireless technology in industrial networks, *Proceedings of the IEEE*, 93(6), 1130–1151, June 2005.
2. D. Dzung, C. Apneseth, J.-E. Frey, and J. Endresen, Design and implementation of a real-time wireless sensor/actuator communication system, *ETFA'2005—10th IEEE International Conference on Emerging Technologies and Factory Automation*, September 19–22, 2005, Catania, Italy.
3. J.-E. Frey, G. Scheible, and A. Kreitz, Unplugged but connected, *ABB Review*, 3,4, 70–73, 65–68, 2005.
4. Bluetooth SIG, *Specification of the Bluetooth System*, Version 1.1, February 22, 2001.

5. European Telecommunication Standards Institute, Digital cellular telecommunication system (Phase 2); Multiplexing and multiple accesses on the radio path (GSM 05.02). ETS 300 574, August 1999.
6. A.A. Shaar and P.A. Davies, A survey of one-coincidence sequences for frequency-hopped spread spectrum systems, *IEE Proceedings*, 131(Pt. F .7), 719–726, December 1984.
7. IEEE Standard 802.15.2, IEEE recommended practice for information technology, Part 15.2: Co-existence of wireless personal area networks with other wireless devices operating in unlicensed frequency bands, IEEE Std. 802.15.2, 2003.
8. G. Scheible, Wireless energy autonomous systems: Industrial use? *Sensoren und Messysteme VDE/IEEE Conference*, March 11–12, 2002, Ludwigsburg, Germany.
9. J.A. Paradiso and T. Starner, Energy scavenging for mobile and wireless electronics, pervasive computing, *IEEE CS and IEEE ComSoc*, 4(1), 18–27, 2005.
10. International Commission on Non-Ionizing Radiation Protection (ICNIRP), Guidelines for limiting exposure to time-varying electric, magnetic, and electromagnetic fields (up to 300 GHz), *Health Physics*, 74(4), 494–522, 1998.
11. J. Weczerek, *Drahtlose Übertragung von Steuersignalen in der Automatisierung mittels Bluetooth Wireless Technologies Kongress 2004*, Mesago Sindelfingen, Germany, 2004.
12. S.P. Karanam, H. Trsek, and J. Jasperneite, Potential of the HCCA scheme defined in IEEE802.11e for QoS enabled industrial wireless networks, *6th IEEE International Workshop on Factory Communication Systems*, June 28–30, 2006, Torino, Italy.
13. L. Lo Bello, Communication techniques and architectures for Bluetooth networks in industrial scenarios, *ETFA'2005—10th IEEE International Conference on Emerging Technologies and Factory Automation*, September 19–22, 2005, Catania, Italy.
14. M.L. Mathiesen, R. Indergaard, H. Vefling, and N. Aakvaag, Trial implementation of a wireless human machine Interface to field devices, *ETFA 2006—11th IEEE International Conference on Emerging Technologies and Factory Automation*, September 20–22, 2006, Prague, Czech Republic.
15. F. Schmidt, Batterielose Funksensoren, *11. ITG/GMA Fachtagung, Sensoren und Mess-Systeme;* März 11–12, 2002, Ludwigsburg, Germany.
16. M. Kraetzig, G. Scheible, and R. Hüppe, So erreicht man Koexistenz–Ergebnisse einer firmenübergreifenden Studie des ZVEI, *VDI/GMA Fachtagung Wireless Automation*, 26–28 February 2008, Berlin, Germany.
17. VDI/VDE Guideline 2185, Radio based communication in industrial automation; Issue 09/2007, VDI/VDE-Gesellschaft Mess- und Automatisierungs-technik.
18. C. Dupler and H. Forbes, Wireless technology for the discrete industries, *12th Annual Orlando Forum—Winning Strategies and Best Practices for Global Manufacturers*, Orlando, Florida, February 4–7, 2008, ARC, www.arcweb.com

V

Networked Embedded Systems in Building Automation and Control

29 Data Communications for Distributed Building Automation
Wolfgang Kastner and Georg Neugschwandtner **29**-1
Introduction • Building Services • Building Automation and Its Benefits • Application Modeling • Open Standards • Conclusion and Outlook

29

Data Communications for Distributed Building Automation

29.1 Introduction **29**-1
29.2 Building Services **29**-2
29.3 Building Automation and Its Benefits **29**-3
Integration • Automation and Control
29.4 Application Modeling **29**-6
Data Communication Services • Communication Characteristics and Strategies • Distributing Functions and Services
29.5 Open Standards **29**-16
Plant Room and Field Level Subsystems • BACnet • LonWorks • KNX • IEEE 802.15.4 and ZigBee • Web Services • Home Automation • Standardization
29.6 Conclusion and Outlook **29**-32
References **29**-33

Wolfgang Kastner
Vienna University of Technology

Georg Neugschwandtner
Vienna University of Technology

29.1 Introduction

Building automation is concerned with monitoring and control of *building services* equipment. This specifically includes heating, ventilation, and air-conditioning (HVAC) systems as well as devices for lighting and shading. However, integration with systems from other building disciplines, such as fire safety, is also of relevance. Large spaces need to be covered and centralized (and remote) access to all data is important. The requirements of this domain are best fulfilled by distributed embedded systems. In the following pages, requirements, approaches, and standards for data communication within such systems are discussed.

First, an overview of building services is given to describe the background of automation efforts in this domain. Then, the benefits of automation are presented, including the pros and cons of system integration and separation. Next, common concepts for modeling building automation applications are introduced. This model is then extended to accommodate interoperable distribution of functions over a network. Required communication services and communication paradigms are presented, followed by quality of service aspects and appropriate mechanisms. A discussion of implementation considerations concludes the first part.

The second part focuses on open protocol standards of key relevance in the building automation domain. Besides a discussion of the three most popular representatives (BACnet, LonWorks, KNX), it also includes information on the wireless world (ZigBee) and Web services (WS) based specifications. Finally, a quick glance is cast on home automation and an outlook is given.

29.2 Building Services

Buildings should provide supportive conditions for people to work and relax. This means they will usually be tuned toward human comfort parameters. Sometimes, zones or entire buildings are optimized for the particular demands of machines, processes, or goods, which may differ from human comfort. In any case, the environment needs to be safe, secure, and provide the necessary infrastructure, including supply/disposal, communication/data exchange, and transportation. These requirements vary significantly depending on the purpose of the building.

Buildings fulfill these demands through appropriate design of building structure and technical infrastructure, the latter being known as building services. For example, ventilation can be achieved through opening windows (a structural design measure) or forced ventilation (a mechanical building service).

Building services include "passive" installations for transport of energy, information, and materials (such as power distribution, waste water disposal, and data networks) as well as controllable, "active" systems such as HVAC. Systems do not, however, clearly fall into one of these categories. For example, water supply may include pressurization pumps, and power distribution may be extended with switchgear monitoring, power factor monitoring, or on-site cogeneration.

Different building types will have different requirements regarding presence and performance of these services. Table 29.1 highlights examples grouped by building disciplines. For a comprehensive reference on building systems refer to [1].

While the permissible environmental conditions for goods, machinery, and processes are usually clearly specified, ensuring human comfort is a more complex affair. For example, thermal comfort does not only depend on air temperature, but also air humidity, air flow, and radiant temperature. Moreover, the level of physical activity, the clothing worn, and individual as well as cultural preferences have to be taken into account. Much in the same way, one and the same amount of air flow can be perceived as a pleasant breeze as well as a draft depending on thermal sensation. Also, the amount of control available to individuals has an influence on whether they will consider otherwise identical conditions as comfortable or not. This for instance applies to the ability to open windows and having control over air delivery devices.

Not all sections of a building can (or need to) be treated equally with respect to environmental conditioning. As an example, for access spaces like stairways, thermal comfort parameters are relaxed. Also, the sunlit south side of a building may require different treatment than the one facing north. Therefore, and for reasons of manageability in large complexes, buildings are split into control zones. With room control, every room forms a zone of its own. Conditions can then be optimized for taste or presence.

Typical HVAC equipment includes fan coil units, VAV (variable air volume) boxes, and radiators. Valve and damper actuators are typical control elements. Fans are controlled in discrete steps as well as continuously by way of variable frequency drives. Central plants (such as air handling units, boilers,

TABLE 29.1 Building Disciplines and Example Systems

HVAC	Thermal comfort and air conditioning: mechanical ventilation, air humidification, free convection heating/cooling
Visual comfort	Artificial lighting, shading (motorized blinds/shutters)
Safety	Fire alarm, gas alarm, water leak detection, emergency sound systems, emergency lighting, CCTV (closed circuit television)
Security	Intrusion alarm, access control, CCTV, guard tour patrol systems
Transportation	Elevators, escalators, conveyor belts
Communication technology	LAN, private branch exchanges; voice intercom, public address/audio distribution and sound reinforcement systems
Supply and disposal	Power distribution, fresh water, hot water, waste water, gas, switchgear, pumps
Application specific	Clock systems, flextime systems; audiovisual systems (conference rooms, auditoriums, digital signage); kitchen equipment, laundry/cleaning equipment, laboratory equipment

or chillers) in particular come packaged with customized controllers. Typical sensors measure flow and space temperatures, humidity, air quality, channel air pressure, or presence.

Systems for visual comfort have the task of providing just the right amount of light for occupants. Artificial light can be modulated to fill in for missing daylight. Shading limits the amount of daylight which enters the interior to avoid excessive light intensity and glare. Typical actuators and sensors involved are load switches, incandescent dimmers, controllable ballasts, sunblind drives, and presence detectors, luxmeters, and anemometers (for protection of outside blinds). Recently, electrochromic windows have become available commercially, whose translucence is continuously adjustable by applying a low voltage.

In safety and security alarm systems, alarm conditions have to be detected and passed on to appropriate receiving instances. This includes local alarms as well as automatically alerting an appropriate intervention force. Precisely distinguishing nonalarm from alarm situations is essential. Example sensors are motion and glass break sensors from the security domain, water sensors for false floors from the property safety domain, and smoke detectors, heat detectors, and gas sensors from the life safety domain. Alarms are announced by klaxons or playback of prerecorded evacuation messages. Emergency lighting is also related to this field. Related areas of importance with a certain overlap with the HVAC domain are fire damper monitoring and control as well as active smoke control.

29.3 Building Automation and Its Benefits

The task of building automation systems (BAS) is to provide automatic feedback control as well as central (and remote) monitoring and access to building systems. Automatic control is typically performed in the HVAC and lighting domains.

By applying optimized strategies in these highly energy intensive areas, considerable savings in operational cost can be made. Good HVAC control strategies can optimize the consumption of primary energy by capitalizing on information about thermal comfort conditions as well as properties of the building structure (high or low thermal inertia) and systems. Comprehensive sensor data (automated strategies may also automatically take air quality into account) and provisions for fine-grained control also work toward this goal.

Example functionality related to power-saving operation is optimum start and stop control (automatic shutdown of air-conditioning equipment during nonoffice hours, but starting earlier and stopping sooner to allow for system inertia) and night purge (ventilate with 100% outdoor air at night). The amount of primary energy to be converted in boilers and chillers can be automatically adjusted based on calendar data or in response to what is required by the individual zones.

Demand control of lighting systems can also significantly contribute to energy saving. In functional buildings (e.g., office buildings, hospitals, department stores), these benefits—calculated over the entire building life cycle—will easily justify the required investments. Only a minor fraction (typically about 20%) of the total building life-cycle cost accrues from the construction phase. Positive environmental effects are another aspect to consider, and legislation requiring energy efficient buildings is increasingly found.

Also, increased comfort must be mentioned as a benefit that cannot be easily quantified, but should not be underestimated. Increased comfort will improve workforce motivation and productivity, and will raise the perceived property value. This appeals to tenants concerned with their image and allows building owners and tenants to set themselves apart from the competition by offering higher value.

Obviously, central access also reduces cost. This allows problems to be narrowed down more easily and quickly. Preventive as well as corrective measures can be planned more efficiently. Remote access is particularly useful when building sites are spread over a large area. Alert messages may be forwarded to the operator via cellular text message gateways or electronic mail. Travel time and costs are avoided and faults can, ideally, be resolved without being on site. Optimum scheduling of

preventive (or even predictive) maintenance based on automatic equipment monitoring and service hour metering are possible.

Moreover, collecting (and allocating) energy consumption data with high resolution would not be feasible without central data acquisition. The resulting increase in cost transparency finally allows more precise controlling measures to be put in place.

This is not limited to HVAC and lighting. While other systems often stay stand-alone from a automatic control point of view, they can be integrated into a single building management system (BMS). Recognizing the head start of BAS with regard to central data acquisition, control, and presentation, they provide the natural base for the successive integration of other systems.

A common way of accessing all systems, supported by tailor made user controls, allows changes during regular operation to be applied without special training, despite the controlled equipment being technically complex.

29.3.1 Integration

"System integration" is an everyday reality (and necessity) even for BAS that solely target the HVAC domain. System parts must be combined to form the HVAC control system, and devices must be configured to interact properly. Often, cabling, sensors/actuators (field devices), cabinets, controllers (for those pieces of equipment that do not have their own), and management infrastructure are tendered separately and provided by different specialist contractors (as is, of course, the HVAC equipment itself).

This division is even more pronounced between building disciplines, since they have evolved separately. Consequently, their respective automation systems are as well still entirely separate in most buildings today. However, information exchange between systems of different domains undeniably provides benefits. Actually, such cross-domain integration is indispensable for intelligent buildings [2].

For example, in case of a fire alarm, the HVAC system could enter a special mode for smoke extraction, elevators could automatically stop loaded cabins at the next floor level and shut down, and the flextime system could print out a list of all people having checked in. Another scenario would be lighting a pathway through the building and controlling elevators based on card access control at the gate.

A key area of cross-domain integration is room automation. For example, window blinds have considerable impact on HVAC control strategy, as incident solar radiation causes an increase in air temperature as well as in immediate human thermal sensation. Automatically shutting the blinds on the sunlit side of a building can significantly decrease the energy consumption for cooling purposes.

A (computer aided) facility management system that has direct access to data from the BAS becomes an even more efficient tool for, among others, accounting and controlling purposes. For example, consider cost allocation for climate control and lighting using live metering data.

As another example of BAS integration with IT systems, consider conference rooms to be air-conditioned only (and automatically) when booked. Also, hotel management systems can automatically adjust HVAC operation depending on whether a room is currently rented or vacant.

Yet, in all cases, the benefits reached by tighter integration come with a drawback. In an integrated system, examining subsets of system functionality in an isolated manner becomes more difficult. This introduces additional challenges in fault analysis and debugging as well as functionality assessment. Additionally, if multiple contractors are working on a single integrated system, problems in determining liability may arise.

This assessment problem is of special concern where life safety is involved. For this reason, fire alarm systems traditionally have been kept completely separate from other building control systems. Although a considerable degree of integration has been achieved in some projects, building codes still often disallow BAS to assume the function of life safety systems. This of course does not

extend to less critical property safety alarms, such as water leakage detection. Similar considerations apply to security systems.

These issues need to be addressed by carefully selecting the points of interaction between the different systems, with the general goal of making the flow of control traceable. Interfaces have to be defined clearly to ensure that no unwanted influence is possible. Typically, systems stay standalone for their core purpose. Only a highly limited number of interaction points are established at the highest system level. However, such "high-level integration" can already yield considerable benefits. For example, consider again the fire alarm example: the information that an alarm condition is present is only a single bit.

29.3.2 Automation and Control

Building automation can be regarded as a special case of process automation, with the process being the building indoor environment (and its closer surroundings). Although some applications, such as shading, will actually involve outdoor sensors and actuators, environmental conditions will typically only be controlled in the interior.

Providing a comfortable environment to humans is a very complex control target. Since some of the factors influencing comfort will by their nature remain unknown to the system,* continuous manual intervention (by occupants and/or privileged operators) is part of routine system operation in home and building automation.

Since HVAC processes involve large (thermal) capacities, system state only changes gradually. Quick transients only have to be detected when optimizing system behavior, if at all. Since plant response times are slow, requirements on controller response times are relaxed compared to industrial control applications. Despite the general absence of high-speed control loops, HVAC control is not without challenges. It has to deal with disturbances which change over time as a function of load, weather conditions, and building occupancy. These influences are of stochastic nature and therefore not exactly predictable, although certain assumptions can be made. A comprehensive introduction to HVAC control is, e.g., provided in [3].

Closed-loop control is barely present in other building systems. Interestingly enough, timing constraints are tightest in certain open-loop control relations (most notably simple light control functions), where the response time is put in relation with the human perception time in the range of a few hundreds of milliseconds.

Definitely, one key challenge in BAS is that large areas need to be covered especially in high-rise buildings or larger building complexes. However, most feedback loops are highly local.

Regarding the required reliability (defined as the probability of a system to perform as designed) and availability (defined as the degree to which a system is operable at any given point in time), demands are moderate as well. The consequences for failing to meet them are merely annoying in the vast majority of cases, although exceptions do exist (e.g., refrigerated warehouses or medical applications).

In case of automation system failures, graceful degradation is expected. Basic functions have to be provided with higher reliability and availability than more complex ones (e.g., examining trend data). Also, control loops should be able to operate independently (i.e., loss of control within one zone or building wing should not affect another).

Requirements in this respect are highly different for the security and safety domain. Here, very high reliability is expected, although certain compromises can be made regarding availability (failures in the system must be detected with high reliability).

* As examples, consider the clothing worn or lighting that is appropriate for an occupant's current mood.

Another challenge is that the domain is highly cost sensitive. Also, systems have to be long-lived, as they are expected to operate over a significant part of the building life cycle. They are required to be "future proof," which favors proven, technologically conservative approaches. Hence, the domain is very slow to accept and adopt new technological developments. Bid invitations often require systems to adhere to international standards, which lengthens the innovation cycle due to the delays inherent to such standardization procedures. Finally, ease-of-use is of significant importance, both for operators and occupants.

29.4 Application Modeling

In the world of process and BAS in particular, *data points* (or simply *points*) are a very common concept. Each data point logically represents a single value from the underlying physical process. Input data points represent sensor measurements, while output data points map to actuators. In addition, data points may exist which do not immediately map to a physically observable value, such as a set point. These are called "soft" or "virtual." In contrast, those which correspond to an aspect of the real world (such as a certain room temperature or the state of a switch) are called "hard" or "physical."

The overall application is defined as functions over these points. The relevant world standard [4] divides these functions into three categories: Input/output (I/O), processing, and management.

I/O functions basically wrap a data point. Thus, they describe the task of actually interacting with the physical world: acquiring environmental and plant data (analog measurements, digital state inputs, pulse counting for metering purposes) and controlling parameters of the physical world by way of building services equipment (switching, setting/positioning).

Processing functions then build upon the I/O functions. Feedback control and control sequences are executed. Thus, these functions describe the required automation function (including higher level strategies such as night purge). In addition, [4] includes functions for building services equipment maintenance such as monitoring of data point limits or run hours in this category.

Management functions form the third category. They describe the capability of a system to collect information for human operators to enable them to analyze and optimize its behavior (e.g., by changing controller parameters or schedules). This specifically includes the accumulation of historical data (trend data) at specified rates (or value increments) and the possibility to generate reports and statistics.

For each category, the system has to display relevant data and allow human interaction where required. For example, measurements need to be displayed to the occupant and/or the system operator, and set points need to be shown and modified. User interface elements will differ according to the type of interaction. As examples, consider manually overriding an output right at the actuator versus examining trend data.

The data flow through the system can be shown in a function block diagram. Function blocks can be data sources (in case of an I/O function which represents an input data point) or data sinks (in case of an I/O function which represents an output data point). They can also transform data (representing a processing function, which is immediately associated with a set of processing rules). Figure 29.1 also includes a virtual datapoint, which could represent a controller setpoint, and a block representing a management function (trending/logging). The blocks that are present and their connections form a directed graph that defines the overall automation application.

For automatic control, nothing more than the present value of a data point is required: to perform its task, a controller does not require information about whether the damper it controls resides in the basement or on the top floor.

However, points are also used as a key abstraction for management purposes. In BMS, they provide the central means to present information from all connected systems to the operator in a unified way.

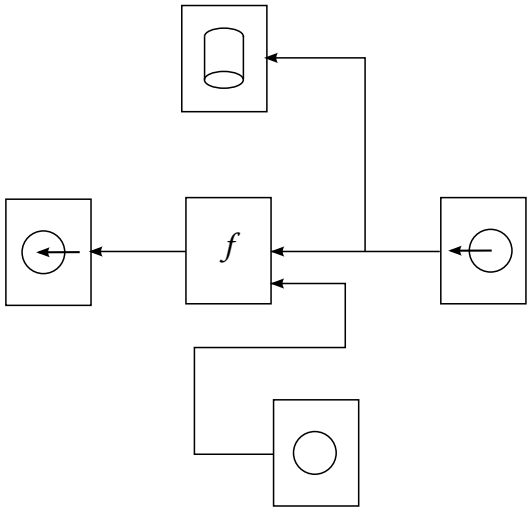


FIGURE 29.1 Data flow between function blocks.

This is when point *meta-data* come into play. These additional attributes are associated with data points besides their main characteristic (the present value) to further qualify the data point value.

A unit attribute adds a semantic meaning to the present value by describing the engineering unit of the value. A precision attribute specifies the smallest increment that can be represented. Attributes such as minimum value, maximum value, and resolution may describe the observable value range of the data point more precisely. The resolution can be the actual resolution of the physical sensor and may be less than the precision.

The location of a point certainly is a key attribute. It is often correlated to a name. Building planners may design the point name space according to geographical aspects, such as building, floor or room and/or according to functional domain aspects, such as air conditioning or heating. An example pattern is Facility/System/Point, e.g., "Depot/Chiller1/FlowTemperature."

If the system allows limit monitoring for a data point, certain bounds can be set on its value. Alarm indicator attributes then show if these bounds have been exceeded. For example, when a temperature limit has been exceeded, the data point can switch from normal mode to alarm mode. This attribute can be persistent so that it can be used to detect alarms also after the value has returned to be in bounds again. Often, a distinction is made between events (any exceptional condition, e.g., that an air filter needs to be replaced) and alarms, which require immediate action. Notifications can therefore be associated a priority. An additional, important concept for data points are point priorities. In building automation applications it is common that output points are associated with multiple functions (control strategies) in parallel, whose outputs can be in conflict. For example, a damper may be closed by the standard control loop. This may be overridden manually to allow ventilation for a meeting after hours. This command will again be defeated when a fire alarm is raised and the damper needs to be closed for fire protection. The priority of the function that is currently controlling the state of a point can be represented as one of its attributes.

29.4.1 Data Communication Services

As will be discussed in following sections, a distributed implementation approach is well suited to BAS. Where system functions require multiple devices to cooperate over the network, suitable communication services need to be present. This relates to semantics as well as quality of service aspects.

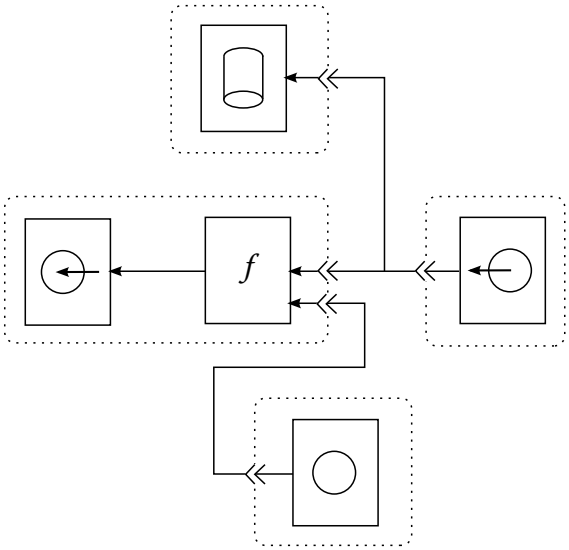


FIGURE 29.2 Data flow over the network.

The most important class of communication services is concerned with the *exchange of process data*. Returning to Figure 29.1, we shall assume (as it is usually done) that the elementary functions represented by the function blocks each reside on a single device. However, each device may host multiple function blocks. The data flow between function blocks hosted by different devices now has to cross the network. It leaves and enters devices via communication endpoints (Figure 29.2).

Establishing the communication links between communication endpoints to set up the proper data flow (at installation time) is known as *binding*. These logical communication links can be entirely different from the physical connections between the nodes.

For a useful binding to be possible, the semantics associated with the communication endpoints at both sides have to match. First of all, this concerns the basic digital value representation (type, such as Integer, Floating point, Boolean, or enumeration types; and encoding, such as unsigned 8-bit integer, or IEEE 754). Additional semantics (e.g., engineering unit, value range, precision) can be added to create complex types, such as one representing "humidity." Finally, for a complete definition of communication endpoint semantics, dynamic aspects must be considered. Communication events (value changes, updates) on a particular endpoint must be associated with aspects of system behavior (e.g., a relative setpoint change or an actuator moving to a new position). This is best done at the level of function blocks.

Standardizing the communication interface of function blocks and their behavior with respect to their individual communication endpoints is a decisive step. First, it enables reuse as planners can construct templates of complex functionality and instantiate them multiple times without repeated engineering effort. Moreover, it is essential for devices which are designed to perform a very specific, commonly used function (such as measuring temperature). As long as bindings can be defined separately from the node application, the latter can be provided by the device manufacturer for efficiency. When defining these *device profiles,* it is especially important to keep in mind that binding them should yield a useful distributed application.

Such profiles may be generic (e.g., analog output) as well as domain specific (e.g., dimming actuator). Generally speaking, less complex profiles are easier to understand and leave less room for error but incur higher engineering effort for system integration because the semantics missing from the profile have to be engineered manually.

This high-level view should be reflected by the data model and services of the network protocol. For the node application programmer, a so-called shared-variable model is particularly convenient. It is a good match for the data flow centric nature of this high-level view and quite naturally allows separation of the node application from its communication bindings. On every change of a specially designated variable (possibly holding the present value of an output data point), the nodes' system software will automatically initiate the necessary message exchange to propagate the updated value to the appropriate receivers. This style of communication is also referred to as "implicit messaging," as opposed to "explicit messaging."

Communication services roughly fall into three categories: process data exchange, management, and engineering/diagnostics. Not all of them are necessarily available within a particular system.

Process data exchange (process communication) comprises aspects of transferring data related to I/O and processing functions. Endpoints may have command/event semantics (at-most-once) or state/value semantics (at-least-once). Command invocations may be associated with priorities. Value transfer may be initiated by the data sink (cyclic polling) or by the data source (likewise in a time-triggered manner, or in response to a change of value).

Communication relationships for process data exchange are static and can be established at system setup time. The addresses of the source and sink(s), respectively, thus will be predefined in the node. This is also known as identified communication.

As an alternative to the client–server paradigm, data sources can use content-based communication, using tags that describe the type of content of their transmissions instead of recipient addresses. In this case, sources do not know their associated sinks (producer–consumer paradigm). This mode of communiation is also identified, since the tags are necessarily static.

Management services support management functions and those processing functions related to building services equipment maintenance. Communication services for monitoring of data points are of particular importance.

Communication relationships are less static than for process communication. Communication is more often initiated on-demand, the communication endpoints chosen ad hoc. Publisher–subscriber-based services which allow data sinks to subscribe with the data source, specifying the criteria of interest (e.g., minimum or maximum limit), are well suited to this purpose. With these services, the list of sinks served by a source is managed dynamically, either by the source itself or a separate publisher entity.

Engineering and diagnostics services support setup and maintenance of the distributed automation system itself. Services must be provided to support the binding process as well as device application downloads (configuration) and setting of application parameters (parametrization). Identification services are useful to designate nodes as targets for subsequent engineering actions by way of manual interaction (e.g., pushing a button on the device that is to receive a program download). Example services related to diagnostics are heartbeats (and their monitoring), which can be used to detect device or cable failures, and counters for failed transmissions.

In addition to manual configuration of bindings, system concepts may include support for devices to provide self-binding capabilities. Usually, the system integrator is responsible that the semantics of bound communication endpoints are compatible. Automatic binding schemes may use standardized identifiers for particular matching combinations of communication endpoints to replace this knowledge. Such an approach necessarily reduces flexibility as it requires a more stringent high-level application model.

Ad hoc operation, as it is typical for engineering and diagnostics services, means that communication relationships are not predefined. Thus, these as well as management services are best supported by services which allow the devices (or communication endpoints) present on the network to be discovered automatically. In addition, networks should be able to provide descriptive information about themselves by providing communication services that enable the attributes to be associated with devices and communication endpoints, e.g., type identifiers (possibly including the associated

function block type), physical location tags, network addresses, serial numbers, or product IDs. Such discovery and self-description services minimize the dependence on external, possibly inaccurate network management databases and enable entirely ad hoc operation (e.g., connecting a generic handheld operator panel for the purpose of troubleshooting).

29.4.2 Communication Characteristics and Strategies

Functional buildings often reach considerable spatial dimensions. Devices implementing I/O functions necessarily need to be distributed for localized control to be possible. However, most feedback loops are highly local. Therefore, a distributed control approach suggests itself, with multiple controllers responsible for locally contained subprocesses. A distributed approach also supports graceful degradation as it helps avoid single points of failure. It allows subsystems to be out of service due to failures or scheduled maintenance without affecting other system parts.

Certainly, distributed systems are harder to design and handle than centralized ones. Yet, the increase in complexity for the overall system will be mitigated when "divide and conquer" is applied properly, with the added benefit of the resulting subsystems being more transparent.

The system functions described differ with regard to their communication requirements. Evidently, the amount of data managed by a single function (or a device implementing that function, respectively) increases from I/O over processing towards management functions. This applies to the spatial as well as the temporal domain. In parallel, the number of devices that implement such functions decreases. The shape of the well-known automation pyramid stems from this fact. I/O functions cover the present values of single data points. Processing functions operate on multiple I/Os in their vicinity, and may involve aspects of timing. Management functions aggregate data from all over the system over long periods of time.

Process values can typically be represented in a compact way (considering a single value at a single point in time). Moreover, building environmental control does not require high-frequency control loops. Also, event load from stochastic sources (e.g., light switches) is low. Devices implementing I/O and processing functions thus only require a low data rate at the network interface.

Since the spatial locality of control relationships is high, networks can be divided into segments where accumulated throughput will still be low. Thus, within each of these segments, low network data rates will be sufficient. Response times need to be acceptable for I/O functions.

Management functions, however, require access to data from all segments. Therefore, data traffic from the entire system will pass through the network segment that the devices implementing these functions are connected to. In larger systems, the amount of traffic thus accumulated is considerable: Thousands of data points are a typical order of magnitude. Therefore, higher network bandwidth is required in this place.

As a general rule, management and engineering services are more demanding in terms of network throughput to provide acceptable speeds for larger block data transfers like trend logs or application program files, but have lower requirements on timeliness.

Also, data seldom need to be available with full spatial and temporal resolution in real-time at a management workstation. For example, it can be perfectly acceptable for the state of a luminaire to be updated with the central monitoring application every 2 min only, since timely response to occupants' requests is ensured by the local controller.

Therefore, a hierarchy of network segments is deployed, with individual segments tuned to these different requirements. For segments at the bottom of the pyramid, cost efficiency, robustness, and easy installation (which again helps save costs) are of particular relevance.

Wiring can be significantly simplified when a network supports free topology.* Supplying power to the nodes over the network cable (also known as link power) both saves additional power wires

* One can think of free topology as increasing the stub length in a bus topology until the bus character disappears.

and allows compact, inexpensive device power supplies. Retrofitting projects will also profit from the ability to use power-line communication. Wireless technologies excel for low-power devices (such as sensors in particular), inaccessible or hazardous areas or when special aesthetical requirements are present. However, replacing batteries can become infeasible when a large number of nodes is installed. Therefore, energy scavenging techniques are particularly attractive. As an example, transmitters that generate the energy required to signal a button press by radio communication from the button press itself (using piezoelectric materials) are commercially available, as are entirely solar powered modules.

The network should also provide appropriate immunity against electromagnetic interference. Robustness in this respect is again desirable especially at the bottom of the pyramid, where cables are laid in the immediate vicinity of the mains wiring. Apart from this, the environment of BA networks is not particularly noisy, especially in office buildings.

Network segmentation requires appropriate protocol support. Network protocols need to provide hierarchical subdivisions and appropriate address spaces. Larger installations will run into thousands of meters of network span as well as thousands of nodes. Networks should also be able to transparently include wide-area connections, possibly dial-on-demand. Automatic routing should be possible. However, this is not a strict requirement since process data exchange is static. Mesh routing increases robustness as it makes use of redundant communication paths. While wired segments are usually not designed redundant, given the moderate availability requirements, this feature becomes relevant for open media such as wireless segments (which, by the nature of the medium used, offer multiple physical links in parallel).

As regards further network layer functions, protocol support for multicast relationships can contribute to efficiency as one data source will often be bound to multiple sinks. Multicast support also may facilitate obtaining set coherence in the time domain on a broadcast medium (which, for example, can be of interest when switching groups of luminaires). Broadcasts are needed to support functions like device or service discovery and clock synchronization.

On the transport layer, engineering services are supported by the availability of reliable point-to-point connections. For process data exchange, best-effort semantics are often sufficient since end-to-end confirmations (i.e., feedback reflecting the actual state of building services equipment) have to be provided anyway.

Reviewing the key requirements, communications efficiency (since it translates into cost efficiency) and moderate dependability (robustness, reliability, availability, safety) are required. Dependability not only translates into the ability of the network to detect transmission errors and recover from any such error or other equipment failure but also involves meeting time constraints. For all timeliness requirements (periodicity, jitter, response time, freshness/promptness, time coherence; cf. [5]), moderate requirements apply. It is sufficient even for more demanding applications to be able to state certain upper bounds on transmission delays.

Therefore, peer-to-peer, event-driven communication schemes are commonly found. While peer-to-peer semantics support graceful degradation, event-driven schemes are highly efficient (especially given the low event load).

Medium access control using deterministic CSMA (Carrier Sense, Multiple Access) variants, possibly supporting frame prioritization, will allow efficient use of the "raw" throughput capacity available as well as fulfill time constraints.

If a network segment is only used for I/O functions (with communication initiated by a single controller node only), master–slave approaches are common. They provide better efficiency due to the nonexistence of arbitration overhead.

For life-safety and security applications, guaranteed performance is mandatory (although still with relaxed timing requirements). Since these applications center around cyclic polling of numerous sensors, time-driven master–slave approaches are suitable.

Last, but not least, security is also an important aspect in terms of quality of service. Securing connection points for remote access over public networks is of particular importance. However, even

if remote access is not possible (or sufficiently secure), the possibility of local attacks must not be neglected. This is especially true when segments are run through publicly accessible spaces or use open media. In any case, the security focus is on authentication. For example, it is usually not a secret that a door was unlocked; however, only a trusted entity should be able to do so.

29.4.3 Distributing Functions and Services

The typical structure of a present-day BAS is shown in Figure 29.3. So-called direct digital control (DDC) stations* are assigned the control of individual parts of the building (zones). DDCs need not be freely programmable. Commonly used functions (such as, specifically, room control) are also often implemented by application-specific controllers. Often, controllers are arranged in a hierarchy.

The DDC station controls actuators, such as fans or valves for hot water and coolant, in response to sensor values and set points (provided from a central location or via local operating panels). Sensors and actuators are most often directly connected to controllers via standard interfaces (like voltage-free contacts, 0–10 V or 4–20 mA), sometimes also via a fieldbus.

The data provided by the DDC stations (sensor values, subsystem status information) are recorded by a central server station. The server station also provides a point of integration of these process data with data from other, stand-alone building systems such as safety alarm systems or access control systems, for unified visualization as well as alarm management. Also, sequences that involve different systems can be automated. In addition, dedicated special systems (DSS) may also connect to DDCs. For instance, a fire/security panel could put HVAC controllers into smoke extraction mode when a fire alarm is raised on its line. The user interface is displayed on office workstations. For remote access to the server, Internet (IP)-based technologies are commonly used.

In the traditional view, the field, automation, and management levels refer to particular classes of devices. Roughly, freely programmable DDC stations (mounted in central cabinets) are considered automation level devices; sensors, actuators, application-specific controllers, room units and room controllers are typically considered as belonging to the field level; and the server station and workstations are considered the management level.

While DDC stations are typically linked to each other and to the server station via a specialized field bus,† workstations access the server via the office network. Thus, the traditional system architecture comprises three (or two, if sensors and actuators are connected directly instead of via a field network) networks using different media and protocols. For a long time, this was a sensible choice since devices at the bottom of the pyramid could not handle the complexity of a protocol for management services (neither the functions themselves), and office networks could not handle timeliness requirements.

However, progress in computer engineering has led to new perspectives. Increasing processing power and available memory, with overall size and cost of end devices decreasing simultaneously, pave the way for so-called intelligent field devices. Sensors, actuators, and room units (control panels with sensor function) now take over automation functions that were previously left to DDC stations. Also, DDC stations are increasingly being equipped with powerful network interfaces and are used for management functions.

If devices implement a mix of appropriate functionality from all three levels, network architectures have to cater to this by providing an appropriate mix of services. Thus, the most flexible solution is to use a single protocol throughout the system.

---

* A DDC station can be compared to a PLC (programmable logic controller). Its instruction set and process interface equipment connections are especially designed for feedback control tasks.

† Data exchange between DDCs is sometimes referred to as *horizontal communication*. Communication between DDCs and the server station is then called *vertical*.

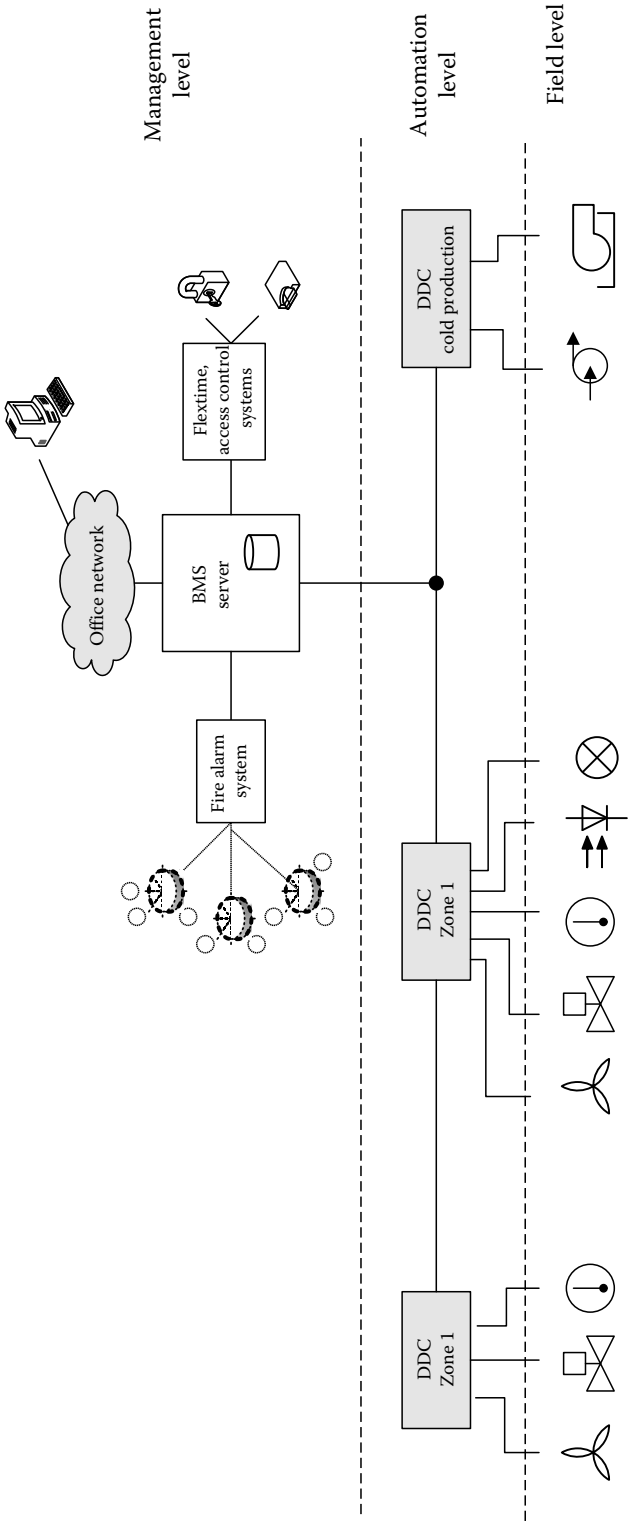


FIGURE 29.3 Building automation: layer model.

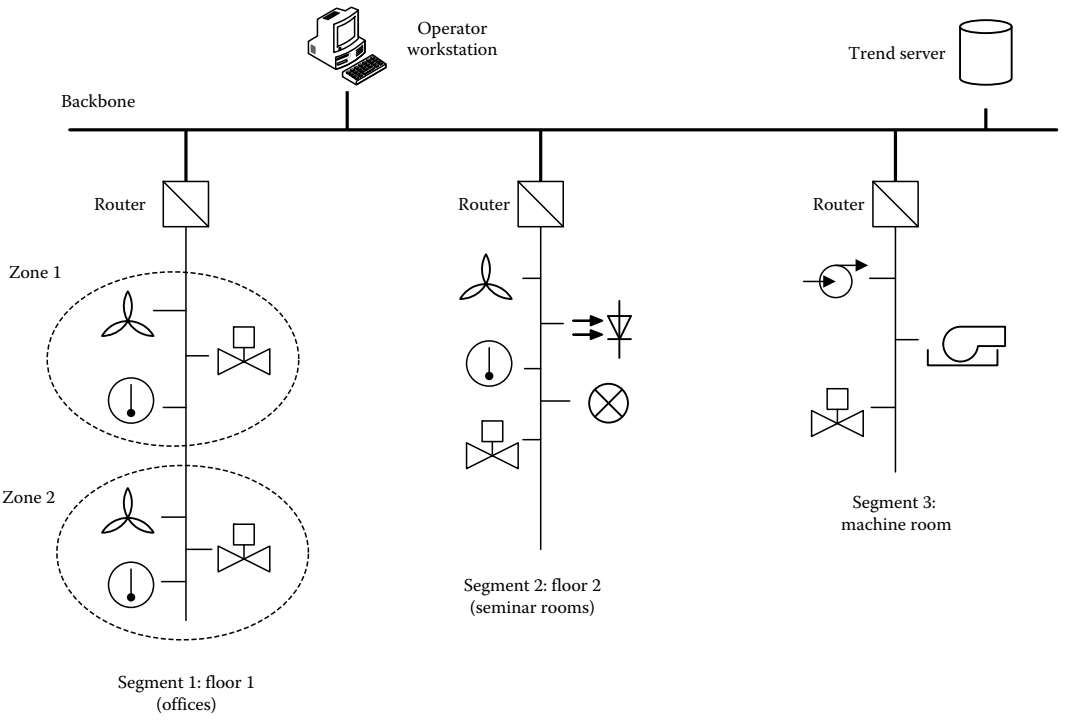


FIGURE 29.4 Automation network and backbone.

Even if this has become commercial reality today, requirements at the top and bottom of the pyramid remain diverse. In particular, this concerns throughput and ease of installation.

Therefore, a two-tier architecture has become popular where *fieldbus segments* are interconnected by a high-performance network *backbone* (Figure 29.4). While they can share most of the protocol stack, different communication media are deployed for cost efficiency.

A typical building network infrastructure consists of independent fieldbus segments for each floor, which connect sensors and actuators at the room level. The backbone channel that connects them may also span building complexes. Plant networks may use a separate controller network, although DDC stations will often connect to the backbone directly.

The data rate of the fieldbus segments remains geared towards the requirements of process data exchange. Although especially during configuration and commissioning (program download) large amounts of data need to be transferred, this occurs seldom compared to regular operation. Moreover, it is not time critical. Therefore, longer transfer times are acceptable.

As a parallel trend, strongly domain specific busses are emerging. In contrast to those protocols which are built around the concept of using a single protocol throughout the system, these protocols are not intended for multiple application domains. They replace the interfaces that were (and are) traditionally used to connect field devices to controllers (voltage-free contacts, 1–10 V, Pt100 …).

They basically contain services to access a mix of I/O functions, together with highly application-specific processing functions. Their application layer design does not suggest significant relocation of automation functions into field devices. Therefore, these busses can only be used together with domain-specific controllers, gateways, or (future) DDC stations equipped with appropriate interfaces. In return, focusing on a particular application domain and a minimum set of functions allows additional functionality (such as "Plug and Play") to be realized, while keeping resource requirements low.

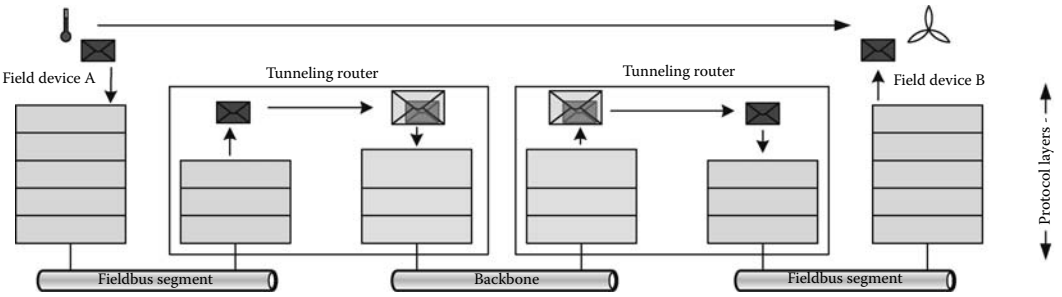


FIGURE 29.5 Tunneling router.

For the backbone, IP (Internet Protocol) has found widespread acceptance.* Although IP networks cannot fulfill the quality of service requirements of more demanding control applications yet since delay cannot be fully controlled, this does not restrict their use in building automation due to the moderate requirements present.

As a rule, however, the network layers of existing fieldbus protocols cannot be adapted for IP. This can be overcome by "tunneling" of data packets. A tunneling router (Figure 29.5) operates entirely transparently to devices on the fieldbus and allows using any network as a backbone. This method also is suited for enabling remote access via the Internet.

IT networking infrastructure has become an integral part of modern buildings. The attempt to leverage this infrastructure for automation purposes is a natural consequence, and the shift towards IP on the BAS backbone makes it a realistic possibility. However, it may still be advisable to deploy separate network infrastructure for exclusive use of the BAS to avoid administrative and performance issues. The benefit of using IP technology remains since inexpensive commercial components can be used. Moreover, a parallel backbone infrastructure can actually be used as a cold standby for the other network, respectively.

A project engineer thus finds a variety of applicable protocols at his disposal, as well as various options regarding network topology and function distribution. He has the choice between using a common protocol throughout the system and only varying the network layer according to the expected traffic load, or connecting subsystems that use specialized protocols via gateways. He could use DDC stations to implement automation functions, or use intelligent field devices instead. However, these decisions are not entirely independent. For instance, in a mixed protocol environment, it is advisable to colocate controller functionality in gateway devices, since both functions require parameterization.

The requirements and conditions of each particular project ultimately determine which of the different combinations will be ideal. If, for instance, an application domain specific bus is used, its services and objects need to be converted for use within the associated DDC station or the BMS. This makes integrated configuration and setup difficult. However, the hardware cost will typically be lower than for a domain neutral system. The question whether this reduction can offset the extra configuration effort can only be answered on a project-by-project basis.

However, the clear boundary that is established by a gateway may actually be a desirable property since it helps in localizing sources of trouble. This is especially welcome if the systems joined by the gateway are provided by different contractors. Therefore, it is not uncommon to see the use of two tiered architectures with incompatible protocols in practice.

* However, it is not expected to become a cost-efficient solution for field devices in the near future.

TABLE 29.2 Field, Automation, and Management Level

Field	Interact with environment	Obtain and adapt measurements; switching, setting, positioning
Automation	Autonomous processes	Open-loop (logical interconnections) and closed-loop control
Management	Supervise, plan, adjust (global preferences)	Presentation, notification, logging; configuration

Owing to these many degrees of freedom, the relevant part of the BA world standard [6] also refrains from specifying a particular system architecture. However, the division into field, automation and management levels remains useful, provided that it is understood as a purely functional classification (instead of classes of device hardware).* For this reason, the generic system architecture of [6] also remains aligned with it. The three-level functional classification is summarized in Table 29.2.

29.5 Open Standards

The field of building automation has been dominated by a plethora of proprietary solutions for a long time. This may be due to the fact that custom solutions are possible at relatively low cost, given the moderate performance requirements involved.

Gateways can certainly be used to gain access to and integrate all kinds of systems, including proprietary ones. But even when all requirements are met at the time of construction, the long life cycle of a building and its services must be considered. BAS have high life expectancy and need to be capable of continued evolution. A rigid system that solely satisfies the demands identified at design time often makes future extensions or tighter integration impossible. Only an open system can guarantee that one does not find oneself tied to the contractor that originally installed the system when extension or alteration is required. Unlike gateways, which need only expose data points defined at contract time, open systems are indeed future proof.

Customers are recognizing that escaping vendor lock-in can significantly lower the total cost of ownership. Moreover, open systems allow integrators to mix and match devices from different manufacturers for increased performance and cost efficiency. This would not be feasible using gateways. Thus, pushed by market demand for open systems, even market leaders are abandoning proprietary designs. Still, this does not mean that gateways will fall from use any time soon. They may still connect subsystems using different open protocol standards best suited to a particular task.

Very different notions exist concerning the meaning of the word "open" in this context. For the purposes of this discussion, the main criterion is whether a system can be repaired, modified, and extended by everyone with the necessary basic qualifications without having to rely on the original manufacturer. This requires that access to the specifications and implementation of the relevant interfaces and protocols is possible at reasonable and nondiscriminatory (RAND) conditions. Certain fees may be involved, however.

The effort required to engineer a system that uses open standards can easily be higher than for a proprietary system. This can especially be the case when the system is built from a *heterogeneous* mix of components from different manufacturers, since specifications typically leave a certain degree of freedom—resulting in additional parameters that have to be aligned to achieve interoperability and provide the end user with a *homogeneous* system. Therefore, the benefits of open systems are not free. The reduction in life-cycle cost thanks to the flexibility gained, however, is generally considered to offset the initial additional engineering (and, possibly, hardware) cost.

* This nontrivial relationship between functional and operational architecture is known from the OSI layer model.

A data communication standard may be confined to physical characteristics, defining aspects like the medium used or network topology. As a prime example for such a standard, EIA-485 is highly popular for BA control networks. It provides the base for several open and seemingly countless proprietary protocols.

While the protocol stack itself determines important characteristics (such as the available address space or communication paradigms), standardized device profiles are equally essential for achieving interoperability. For implementing devices and systems based on the standard, standard hardware components and commissioning tools may be available. Other aspects of interest include which kind of organization is responsible for development and maintenance of the standard and potential compliance certification procedures.

In the following, a number of open data communication standards that are relevant in the field of building automation are introduced. BACnet, LonWorks, and KNX have achieved considerable significance in the worldwide market (in case of BACnet and LonWorks) or in the European market (in the case of KNX) and are often chosen by both customers and system integrators for complete system solutions. While no such discussion would be complete without mentioning them, a couple of other standards, both established and new, further contribute to the picture and will be covered as well.

29.5.1 Plant Room and Field Level Subsystems

For communication with plant controllers and certain complex field devices (such as variable frequency drives for fans and pumps in particular), fieldbus protocols which are well established in factory and process automation (e.g., Interbus, CAN-based protocols such as Devicenet or CANOpen, and Profibus DP) are occasionally used in building automation as well. However, they are very seldom used system-wide. A Profibus FMS profile for building automation existed but, like Profibus FMS at large, has fallen from use entirely.

Although of similar heritage, Modbus has gained comparably significant acceptance in BA. It is also sometimes used at the room level. Designed by Modicon for their PLCs and published in 1979, it is now maintained by the user and manufacturer association Modbus-IDA. Implementation of the Modbus protocol is license-free, which makes it especially interesting for integration and interfacing between BAS and other systems. Not limited to the HVAC domain, Modbus is also present in systems and devices belonging to other building disciplines, such as alarm systems.

The Modbus application layer is basically confined to reading and writing of Boolean and 16-bit integer register values using a simple request/response protocol. This yields a high degree of flexibility in integration, but causes high engineering effort for coordinating the remaining required semantics. Modbus supports serial communication using a simple master–slave protocol over EIA-485. A total of 247 different slaves can be addressed. The typical data rate is 19.200 bps. A mode of transmission over TCP/IP is also defined, in which every node can be both client and server.

Another example for an EIA-485-based protocol with a similar focus is Johnson Controls' Metasys N2, which, despite being owned by a single manufacturer, also provides a certain degree of openness.

Open standards have also emerged for communication with very specific classes of BA field devices. The current prime example is DALI (Digital Addressable Lighting Interface), published as an IEC standard [7] and widely accepted for lighting applications. DALI allows control of up to 64 (dimmable) electronic ballasts for fluorescent lamps and intends to replace the traditional 1–10 V interface.

A DALI *loop* can contain up to 64 individually addressable devices. Additionally, each device can be a member of 16 possible groups. Devices can store lighting levels for power-on, system failure and 16 scene values, plus fading times. There are also immediate commands (without store functionality) and commands for status feedback (such as lamp status). Assignment of addresses, group membership and scene values is possible via the bus. Loops can be up to 300 m long, with free topology. The data rate is 2400 bps using a master–slave-based, asynchronous protocol. Since biphase coding

is used, the net data rate is 1200 bps. A message cycle consists of the master sending a request (*in-channel*) consisting of the 7-bit slave address (1 bit is used for selecting between individual and group addresses) and a 9-bit command. A slave (ballast) may then return an 8-bit value (*out-channel*), and the cycle starts over.

For remote meter reading, M-Bus [8] has gained a certain degree of importance in Europe. Its application layer supports various metering applications and includes support for advanced functionality like multiple tariffs. It operates on low-cost twisted-pair (TP) cabling, with the data link layer based on the IEC 870-5-1 / 870-5-2 standard for telecontrol transmission protocols. A serial master-slave protocol with data rates between 300 and 9600 bps is used. A segment can contain up to 250 devices and cover a maximum distance of 1000 m (multiple segments are possible). In the master-to-slave direction, data is transmitted using voltage modulation, while in the reverse direction, current modulation signaling is used.

Also in Europe, SMI (Standard Motor Interface) is emerging as a German multivendor standard for communication with sunblind drives. However, it has remained unpublished to date.

29.5.2 BACnet

BACnet (Building Automation and Control Network) was designed specifically for building automation purposes [9]. It was designed with the task of interconnecting DDC equipment in mind, but covers functions of all levels. Due to its strengths regarding management functions it is often used to complement large LonWorks and KNX/EIB (European Installation Bus) based installations.

First published as a standard in 1995 by ASHRAE (American Society of Heating, Refrigeration and Air Conditioning Engineers) together with ANSI, BACnet was made ISO standard 16484-5 in 2003 [10] and was revised in 2007. BACnet is maintained by ASHRAE SSPC* 135. Implementations are not subject to license fees. BACnet user/manufacturer organizations exist around the world, called BACnet Interest Groups (BIG) and BACnet International (previously the BACnet Manufacturers Association).

In general, BACnet is network agnostic, but explicitly includes a number of network layers to create a "critical mass" for interoperability. Of these, MS-TP (Master–Slave Token-Passing, a BACnet proprietary, RS-485 based 78.4 kbps multidrop protocol) and IP are the most commonly used ones. In United States and Asia, ARCNET† is also popular. The other choices are raw Ethernet, LonTalk, and PTP (point-to-point). PTP, a BACnet proprietary protocol, is designed to support modem dial-up connections via EIA-232. The use of the LonTalk protocol is limited to transporting BACnet messages and does not imply any compatibility with LonMark concepts such as standard network variable types (SNVT). While the original standard already specified tunneling routers (Annex J) for the IP medium, IP only became a first class medium in 1999 (BACnet/IP). BACnet/IP made BACnet devices with direct IP connection a reality.‡ UDP is used as a transport protocol. BACnet/IP specifies a virtual link layer whose most important task is to handle internetwork broadcasts without resorting to Internet group management protocol (IGMP).

BACnet *networks* consist of *segments* of the same media type coupled by repeaters or bridges. BACnet networks are connected by routers to form a BACnet *internetwork*. The topology must not contain loops. Segments need not be connected permanently, for example, those attached via PTP. A BACnet network address consists of a 2-byte BACnet network number and a local address. The local address is specific to the link layer medium, e.g., an IP address for BACnet/IP or a medium access

* Standing Standard Project Committee.

† Unlike Ethernet, ARCNET guarantees deterministic medium access. For BACnet, the cost efficient 156 kbps TP variant is typically preferred.

‡ Before, an Annex J router would have had to be included in every device.

control (MAC) address for LANs, and may be up to 255 bytes in length. BACnet routers route packets based on the network numbers. They are required to be self-learning. Appropriate protocol services for topology discovery are provided.

The BACnet network layer provides an unacknowledged connectionless datagram service. Besides unicasts, broadcasts to any single network or to the entire internetwork are possible. Multicast support (including group assignment) is considered a local matter of the underlying network, and thus typically not present. The application layer adds reliable transfers and optional segmentation. The size of an application layer data unit (APDU) may range from 50 to 1476 octets, depending on the capabilities of the device. Connections involving multiple APDUs are not part of the concept.

The BACnet data model follows an object-oriented approach. Currently, 25 object types are defined. These include simple types representing binary, multistate, and analogue inputs and outputs (physical data points) and values (virtual data points) as well as complex objects for scheduling, averaging, trending, and event reporting. Besides mainly application domain agnostic objects, a "Loop Object" representing PID controllers and objects for life safety systems are to be found as well. The "Device Object" is mandatory for every device. It holds information on device characteristics such as the maximum APDU length accepted and the other objects present.

As an example, Table 29.3 shows the "Binary Output" object and its properties. Some properties are mandatory (R = read-only, W = read/write), support for others is optional (O). The properties "object-identifier," "object-name," and "object-type" are mandatory for every object. The object-identifier uniquely identifies an object within a given BACnet device; the object-identifier of the device object is additionally required to be unique internetwork-wide. The "present-value" property is present in every object representing a data point. The BACnet object model can be extended to include new objects or properties in a compatible way.

The BACnet objects are manipulated via appropriate services, following a client–server model. Currently, 40 services are defined, which are divided into five groups: Alarm and Event, Object Access (property manipulation, supporting multiple priority levels), Remote Device Management, File Access, and Virtual Terminal Services. The latter two are implemented less often today, Internet/Web standards (HTTP/HTML, Telnet, FTP etc.) are preferred instead.

TABLE 29.3 BACnet "Binary Output" Object

Property Identifier	Property Datatype	Conformance Code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Present_Value	BACnetBinaryPV	W
Description	CharacterString	O
Device_Type	CharacterString	O
Status_Flags	BACnetStatusFlags	R
Event_State	BACnetEventState	R
Reliability	BACnetReliability	O
Out_Of_Service	BOOLEAN	R
Polarity	BACnetPolarity	R
Inactive_Text	CharacterString	O
Active_Text	CharacterString	O
Change_Of_State_Time	BACnetDateTime	O
Change_Of_State_Count	Unsigned	O
Time_Of_State_Count_Reset	BACnetDateTime	O
Elapsed_Active_Time	Unsigned32	O
Time_Of_Active_Time_Reset	BACnetDateTime	O
Minimum_Off_Time	Unsigned32	O
Minimum_On_Time	Unsigned32	O
Priority_Array	BACnetPriorityArray	R
Relinquish_Default	BACnetBinaryPV	R
Time_Delay	Unsigned	O
Notification_Class	Unsigned	O
Feedback_Value	BACnetBinaryPV	O
Event_Enable	BACnetEventTransitionBits	O
Acked_Transitions	BACnetEventTransitionBits	O
Notify_Type	BACnetNotifyType	O
Event_Time_Stamps	BACnetARRAY[3] of BACnetTimeStamp	O
Profile_Name	CharacterString	O

The alarm and event handling services provide a wide range of options. Clients can subscribe to change-of-value notifications (with variable increments) of any property. Notifications are also possible for further event types such as a binary or multistate input changing state or an analog input exceeding a given range. Multiple event type algorithms may be applied to a single object, and even property, at the same time. Devices can also send change-of-value reports unsolicited. Events can be assigned different notification classes.

Remote device management services include services for dynamically discovering devices and objects by way of their names or identifiers, discovering the network topology, as well as time synchronization services.

In BACnet, devices have a large degree of freedom regarding which services and protocol features they support. Details about which portions of the standard are implemented in a device are documented in its protocol implementation conformance statement. To enable users to judge more easily whether BACnet devices are interoperable, the standard defines BIBBs (BACnet Interoperability Building Blocks). A BIBB describes certain capabilities in one of the five interoperability areas data sharing, alarm and event management, scheduling, trending, device management, and network management. Each BIBB consists of a client–server pair. This allows specifying precisely whether a device requests a service, is able to answer such a request, or both. A BIBB can also imply the existence of certain objects with certain properties. In addition, device profiles are specified. They describe collections of BIBBs that map to the capabilities of typical BA equipment, such as "building controller," "smart actuator," or "operator workstation."

Interoperability testing and certification is performed by the BACnet testing laboratories (BTL), which are associated with BACnet International and BIG-EU. The BACnet standard is being amended continuously. Various addenda cover clarifications and protocol extensions as well as, e.g., improved protocol security, object types (i.e., profiles) for lighting and security and load control, or ZigBee as a medium.*

29.5.3 LonWorks

LonWorks was initially designed by Echelon Corp. as a generic bus system for automation purposes. Today, LonWorks technology is in widespread use for building automation purposes all over the world. The LonWorks system consists of the LonTalk protocol, a tailor made processor (Neuron chip), and network management software (LNS, LonWorks Network Services). By now, the LonTalk protocol is available as ANSI/EIA-709 [11,12] and EN 14908 [13]. In the following, the term control network protocol (CNP) is used to refer to the standardized communication protocol as in these standards.

CNP supports a variety of different communication media and different wiring topologies. Since it was designed as a generic control network, many protocol parameters are free to choose for the designer. To achieve interoperability (see below), a number of communication *channel* profiles were defined. These still include a variety of TP, powerline (PL), and fiber optic channels. The most popular channel for building automation purposes is the 78.1 kbps free topology TP profile (FT-10), which allows physical segments of up to 500 m using low-cost TP cable. A variant providing link power (LP-10) is also available.

For the backbone, the IP tunneling method standardized in ANSI/EIA-852 [14] (also known as the IP-852 channel or *LonWorks/IP*) has largely replaced the fast TP variant (TP-1250) used before. Both tunneling routers and fully IP-based IP-852 nodes are possible. Channel configuration data including channel membership are managed by a central configuration server on the IP channel.

* BACnet/WS was also published as such an addendum.

For the TP medium, a unique medium access mechanism labeled *predictive p-persistent CSMA* is used. Its key mechanism is that when confirmed multicast services are used, a certain prediction on the future network load (i.e., the confirmations to be expected) can be made. The length of the arbitration phase is modified accordingly. Thus, the rise of the collision ratio with increasing load is mitigated. This helps to insure an acceptable minimum packet rate even under heavy load, unlike in Ethernet-style networks using CSMA/collision detection (CD), where the network load has to be kept well below 50%. At the start of the arbitration phase priority time slots are available for urgent messages.

The entire routable address space of a CNP network is referred to as the *domain.* Domains are identified by an ID whose length can be chosen up to 48 bit corresponding to requirements (as short as possible, since it is included in every frame; as long as necessary to avoid logical interference, especially on open media). A domain can hold up to 255 *subnets* with a maximum of 127 nodes each. Hence, up to 32,385 nodes can be addressed within a single domain. A subnet will usually correspond to a physical channel, although it is both possible for multiple physical channels to be linked into a subnet by bridges or repeaters as well as for multiple subnets to coexist on the same physical segment. Routing is performed between different subnets only. In particular, domain boundaries can be crossed by proxy nodes only (which transfer the information on the application layer). Subnets are usually arranged in a tree hierarchy to allow the use of self-learning routers.

Every domain can host up to 256 multicast groups. Groups can include nodes from any subnet. Broadcasts can be directed to a single subnet or the entire domain. Each node carries a worldwide unique 48-bit identification, the *Node ID*. It can be used for addressing individual nodes for management and configuration purposes, while regular unicast communication is handled through logical subnet and node addresses.

For both unicast and multicast, a reliable transmission mode (*acknowledged*) with end-to-end acknowledgments can be selected. In addition to the "one-shot" *unacknowledged* mode, an *unacknowledged-repeated* mode is provided, where every transmission is automatically repeated a fixed number of times. When acknowledged multicast is used, groups are limited to a maximum of 64 members each. Otherwise, they can contain an arbitrary number of nodes from the domain.

For acknowledged transmissions, a challenge–response authentication mechanism is provided. The challenge consists of enciphering a 64-bit random number using a 48-bit shared secret.

The CNP application layer allows generic application-specific messaging, but offers particular support for the propagation of network variables. Network variables are bound via 14-bit unique identifiers (*selectors*). The management and diagnostic services include querying the content type of the network variables (self-identification), the node status, querying and updating addressing information and network variable bindings, reading and writing memory, device identification, and configuring routers.

Thanks to the open specifications, second source protocol implementations exist besides the Neuron chip. Moreover, variants with integrated transceivers are available. A derivative of ANSI C called Neuron C is used to program the Neuron chips, whereas standard ANSI C can be used to program other controllers. Figure 29.6 shows the classic architecture of a LonWorks node. The controller executes the seven OSI protocol layers and the application program, which interfaces with sensors and actuators connected through the I/O interface.

A variety of installation and management tools are available for CNP networks. The wide majority, however, is based on the LNS (LonWorks Network Operating System) management middleware by Echelon. Besides APIs for commissioning, testing, and maintaining, LNS provides a common project database, avoiding vendor lock-in of these valuable data. For configuration of vendor-specific parameters LNS provides a plug-in interface. Using LNS is not compulsory, but ensures open access to the project data (which represent a considerable value in engineer's work).

Applications exchange data via network variables. These are defined within the nodes' application software. During system configuration, logical links are established between network variables

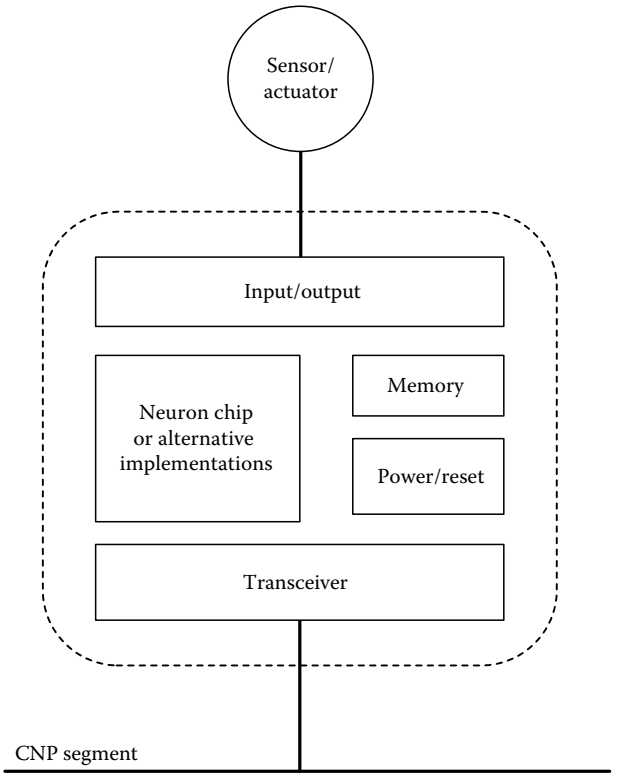


FIGURE 29.6 LonWorks node.

(binding). At runtime, the linked variables are held consistent by the nodes' system software. This is transparent to the application.

No particular compulsory semantics are assigned to network variables. In addition, free-form messages may be exchanged instead. This allows entirely nonopen systems based on LonWorks/LonTalk to be realized, and certain manufacturers do avail themselves of this possibility.

The LonMark Organization [15] defines interoperability guidelines for LonTalk-based devices. These include SNVT and function blocks. A great variety of such blocks has been defined. These include generic control loop elements, management functions (*Data Logger* or *Scheduler*) as well as application domain specific functions, in particular from the HVAC and lighting/shading field (e.g., *VAV Controller* and *Constant Light Controller*). Using these profiles is not compulsory. They are not included in any formal standard.

29.5.4 KNX

KNX [16] can be regarded as an extension to EIB, which is widespread in Europe, in particular in German speaking countries. The KNX protocol is included in the EN 50090 series of standards [17]. Its medium-independent layers were also published within the ISO/IEC 14543 series of standards [18].

EIB was designed to enhance electrical installations in homes and buildings of all sizes by separating the transmission of control information from the traditional main wiring. In 2002, EIB was merged with Batibus and EHS (European Home System) to form KNX. The target of this merger was to create a single European *Home and Building Electronic System (HBES)* standard. Still, the EIB

system technology continues to exist unchanged as a set of profiles within KNX, frequently referred to as KNX/EIB.

The KNX specifications are maintained by Konnex Association, which is also responsible for promotional activities (including a university cooperation program) and the certification of test laboratories and training centers.

Regarding physical media, EIB already provided the choice of dedicated TP cabling and PL transmission as well as a simple form of IP tunneling intended for remote access. RF communication and advanced IP tunneling were added under the KNX umbrella (albeit are not yet published within the context of [17]). The KNX specification also includes additional TP and PL variants which could be used for future devices.

The main KNX/EIB medium is the TP cabling variant now known as KNX TP1. The single TP carries the signal as well as 29V DC link power. Data is transmitted using a balanced base band signal with 9600 bps. TP1 allows free topology wiring with up to 1000 m cable length per physical segment. Up to four segments can be concatenated using bridges (called *line repeaters*), forming a *line*. CAN-like, medium access on TP1 is controlled using CSMA with bit-wise arbitration on message priority and station address. Four priority levels are provided.

KNX RF uses a subband in the 868 MHz frequency band reserved for short-range devices (telecommand, telecontrol, telemetry, and alarms) by European regulatory bodies which is limited by a duty cycle requirement of less than 1%. Particular attention was given to minimizing hardware requirements. To this end, KNX RF does not only support bidirectional communication, but transmit-only devices as well. This reduces cost for simple sensors and switches without status indicators. KNX RF devices communicate peer-to-peer.

EIBnet/IP (or KNXnet/IP, [19]) addresses tunneling over IP networks.* Its core framework supports discovery and self-description of EIBnet/IP devices. It currently accommodates the specialized "Service Protocols" Tunneling and Routing. Actually, both of them follow the tunneling principle, but differ in their primary application focus. EIBnet/IP Tunneling is to provide remote maintenance access to KNX installations in an easy-to-use manner and therefore restricted to point-to-point communication. EIBnet/IP Routing allows the use of an IP backbone to connect multiple KNX (main) lines. Routers using this protocol are designed to work "out-of-the-box" as far as possible. They communicate using UDP multicast. Group management relies on IGMP. No central configuration server is necessary.

As outlined above, the basic building block of a KNX network is the *line*, which holds up to 254 devices in free topology. Following a three-level tree structure, sublines are connected by main lines via routers (termed *line couplers*) to form a zone. Zones can in turn be coupled by a backbone line. Network partitions on open media are typically linked into the topology as a separate line or zone. IP tunneling is typically used for main lines and the backbone, with EIBnet/IP routers acting as couplers. Overall, the network can contain roughly 60,000 devices at maximum (not counting EIBnet/IP specific address space extensions).

Every node in a KNX network is assigned an *individual address* which corresponds to its position within the topological structure of the network (zone/line/device). This address is exclusively used for unicast communication. Reliable connections are possible. Multicast addressing is implemented in the data link layer. For this purpose, nodes are assigned additional nonunique MAC addresses (*group addresses*). The group addressing and propagation mechanism is thus very efficient. Yet, acknowledgements are provided on layer 2 (i.e., within an electrical segment) only. The entire group answers at once, with negative acknowledgements overriding positive ones. This mechanism is

* EIBnet/IP supersedes "plain" EIBnet, which provided tunneling over Ethernet, and the legacy EIBlib/IP (iETS) point-to-point IP tunneling protocol.

limited to the TP medium. If required, end-to-end confirmations must be implemented at the application layer. Group addresses are routed through the whole network. Routers are preprogrammed with the necessary tables. Broadcasts always span the entire network.

KNX uses a shared-variable model to express the functionality of individual nodes and combine them into a working system. Although this model uses state-based semantics, communication remains event-driven. Network-visible variables of a node application are referred to as *group objects*. They can be readable, writable, or both (although the latter is discouraged to better keep track of communication dependencies). Each group of communication objects is assigned a unique group address. This address is used to handle all network traffic pertaining to the shared value in a peer-to-peer manner. Group membership is defined individually for each group object of a node, which can belong to multiple groups.

Usually, data sources will actively publish new values, although a query mechanism is provided as well. Since group addressing is used for these notifications, the publisher–subscriber model applies: the group address is all a node needs to know about its communication partners. Its multicast nature also means, however, that no authentication or authorization can take place this way.

Process data exchange between KNX nodes exclusively uses group addressing. Individual addressing is reserved for client–server style management and engineering communication. System management data like network-binding information or the loaded application program are accessible through the properties of *system interface objects*. In addition, every device can provide any number of *application interface objects* related to the behavior of the user application. On the one hand, their properties can hold application parameters that are normally modified during setup time only. On the other hand, they can contain run-time values normally accessed through group objects. Basic engineering functions like the assignment of individual addresses are handled by dedicated services.

The specification also encompasses standard system components, the most important being the Bus Coupling Units (BCUs). BCUs provide an implementation of the complete network stack and application environment. They can host simple user applications, supporting the use of group objects in a way similar to local variables. Application modules can connect via a standardized 10-pin external interface, which can be configured in a number of ways. Simple application modules such as wall switches may use it for parallel digital I/O or ADC input. More complex user applications will have to use a separate microprocessor since the processing power of the MC68HC05 family microcontroller employed in BCUs is limited. In this case, the application processor can use the PEI for high-level access to the network stack via a serial protocol. As an alternative, TP1-based device designs can opt for the so-called twisted pair-universal asynchronous receiver/transmitter (TP-UART) IC. This IC handles most of the KNX TP1 data link layer. Unlike the transceiver ICs used in BCUs, it relieves the attached host controller from having to deal with network bit timings. These design options are illustrated in Figure 29.7.

For commissioning, diagnosis and maintenance of KNX installations, a single PC-based software tool called ETS (Engineering Tool Software) which can handle every certified KNX product is maintained by KNX Association. KNX devices may support additional setup modes defined by the standard which do not require the use of ETS. First devices implementing them are already commercially available.

Unlike ETS, which allows linking individual communication objects, these modes perform binding at the level of function blocks. Most of these function blocks have only recently found their way into the specification. This may be explained by the fact that EIB has its roots in the field of electrical installation technology, where applications seldom require more than simple switching functions. The KNX function blocks represent application domain-specific functions, many of them from the HVAC field.

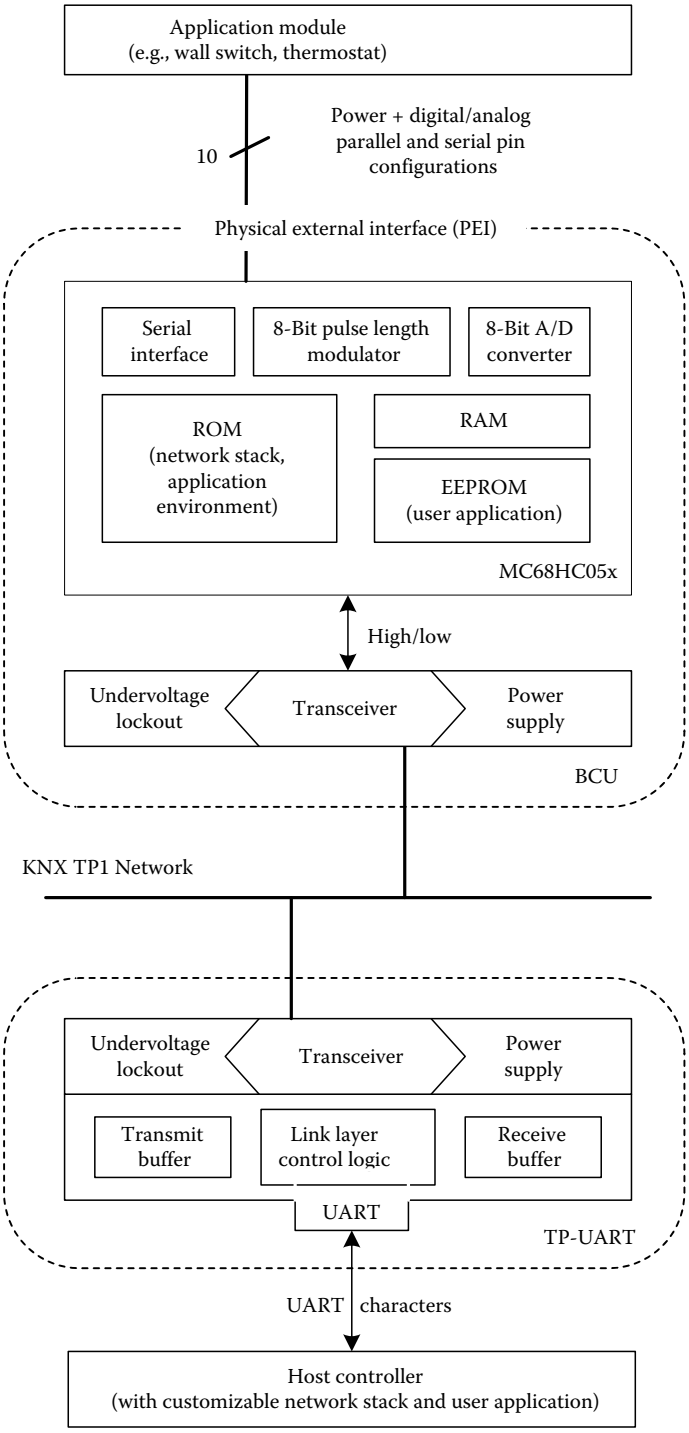


FIGURE 29.7 Typical KNX/EIB node architectures.

29.5.5 IEEE 802.15.4 and ZigBee

Wireless communication has many interesting applications in the field of building (as well as home) automation. For management functions such as log file access for service technicians or presenting user interfaces to occupants on their personal mobile devices, Wireless LAN and Bluetooth are well suited. For purposes of process data exchange, however, they do not provide the required energy efficiency and cost efficiency, especially for wireless sensors.

The focus of IEEE 802.15.4 and ZigBee is to provide general purpose, easy-to-use, and self-organizing wireless communication for low cost and low power embedded devices. While IEEE 802.15.4 defines the physical and the MAC layers, ZigBee defines the layers above. Strictly speaking, IEEE 802.15.4 is therefore an entirely independent protocol, and ZigBee is certainly not the only protocol to build on it—although it is one of the very few published, another notable case being a specification for IPv6 over IEEE 802.15.4 [20]. However, IEEE 802.15.4 and ZigBee are actually not only complementary, but have mutually influenced the development of each other.

29.5.5.1 IEEE 802.15.4

The IEEE 802.15.4 [21] physical layer (PHY) focuses on three frequency bands: 868 MHz, 915 MHz, and 2.4 GHz. The 2.4 GHz band is available license-free almost worldwide. However, this also means it is very crowded, for example, Wireless LAN and Bluetooth use it. BA applications require far less throughput. This enables the use of lower frequencies, which have the advantage of better radio wave propagation with the same amount of power spent. Unfortunately, the frequency ranges for license-free radio communication in the 900 MHz region differ in Europe (863–870 MHz) and United States (902–928 MHz). However, they are close enough to allow a single transceiver design which can be adapted by adjusting the oscillator only.

Although narrower than its U.S. counterpart, the European 863–870 MHz range is particularly attractive since it is well regulated. For example, channel-hogging audio applications such as cordless headphones are not allowed between 868 and 870 MHz, but have their own frequency at 864 MHz. The 868–870 MHz subrange is further subdivided into sections with varying limitations on duty cycle and transmission power. In contrast, devices using the U.S. 902–928 MHz range are only subject to a transmit power limit of 1 W. Therefore, e.g., cordless phones are a major source of interference.

Both frequency bands are supported by IEEE 802.15.4. A variety of modulation schemes with multiple data rates are available. A recent amendment (IEEE 802.15.4a-2007) also specifies ultra-wide band communication in various frequency ranges.

In IEEE 802.15.4, devices are classified as full function devices (FFDs) and reduced function devices (RFD) according to the complexity of the protocol stack. While FFDs can communicate in peer-to-peer fashion, RFDs can only communicate with coordinators, resulting in a star topology.

IEEE 802.15.4 defines two different kinds of personal area networks (PANs): beacon enabled and non-beacon enabled networks. In a beacon enabled network, a superframe structure is used. The superframe is bounded by network beacons which are sent by the PAN coordinator periodically. Between these beacons, the superframe is divided into slots (contention access period) which can be used by the PAN members to communicate using a CSMA/collision avoidance (CA) scheme. Optionally, the PAN coordinator can assign guaranteed time slots (GTS) to devices, providing them with a fixed communication slot. In a non-beacon enabled network, all PAN members can communicate at any time using CSMA-CA. In contrast to other wireless protocols that consider security an application matter, IEEE 802.15.4 specifies different security services despite its low position in the protocol stack. These services rely on AES and include access control, message confidentiality, message integrity, and replay protection.

IEEE 802.15.4 was first published in 2003. In late 2006, 802.15.4b added additional PHY layers as well as MAC improvements (e.g., reduced association time in non-beacon networks). It also included

support for a shared time base, security improvements. GTS support was made optional and more flexibility in the CSMA-CA algorithm was provided.

29.5.5.2 ZigBee

ZigBee [22] adds a network layer enabling self-forming and self-healing mesh networks. For this purpose, it provides appropriate routing services including route discovery and maintenance. It also includes mechanisms for starting, joining, and leaving a network.

Within the ZigBee application layer, the so-called application support sublayer (APS) provides the interface to the network layer. It consists of the APS Data Entity, which provides services for data exchange, and the APS Management Entity, which hosts services for binding and group management.

The services of the APS are used by the so-called application objects (AOs), which actually implement the device application. Each AO forms an independent functional subunit and can be addressed via its endpoint number. A ZigBee device may host up to 240 of them.

The ZigBee Device Object (ZDO) is a special AO residing at endpoint 0 in every ZigBee device. It implements the server entities required for device and network management, for use by both local and remote clients. These include device and service discovery, (self) description, joining networks, binding management, and configuration of security services (e.g., key establishment and authentication).

The communication endpoints of AOs are called *clusters.* The structure of the AOs as well as their clusters is not defined by the core protocol specification. Interoperability between ZigBee devices is enabled by *application profiles,* which define the semantics of messages exchanged between clusters.* Application profiles are each dedicated to a specific application domain. They can be manufacturer specific or public. At the time of writing, the Home Automation profile is the only one published, although more have been announced.

Device descriptions within an application profile specify the clusters that have to be present in a particular AO (and thus, physical device). Examples from the Home Automation profile are *On/Off Light*, *On/Off Light Switch*, and *Pump*.

The ZigBee Cluster Library (which is separate from the core protocol specification) predefines a variety of clusters for use by application profiles. It specifies clusters as collections of commands and attributes, following a client–server pattern. While a single attribute of a cluster represents a single data point to be controlled (e.g., the state of a light), commands are used to manipulate these attributes as well as to initiate actions within the device. In addition, mechanisms for attribute discovery and change reporting (according to defineable criteria) are defined. The cluster library also contains basic data type definitions (e.g., Integer and String). As an example, the *On/Off Light* device from the Home Automation profile contains the (mandatory) server clusters *On/Off*, *Scenes*, and *Groups*, and the optional client cluster *Occupancy Sensing* from the cluster library.

ZigBee security is based on the mechanisms specified in IEEE 802.15.4, but extends them by introducing different keys for end-to-end and network wide security. It also provides mechanisms for key distribution.

The ZigBee standard was first published in 2004. The 2006 release added features such as device groups and multicast, but was not compatible with the original specification. It also removed the original key-value-pair service in favor of the cluster library. The standard update released in 2007 maintains compatibility with ZigBee 2006, adding optional mechanisms for avoiding radio channels with strong interference (frequency agility) and fragmentation. In addition, the "ZigBee-PRO" stack profile offers more features, but only limited compatibility, and has a larger memory footprint.

* In addition, they define network parameters that need to be aligned, such as preferred channels.

29.5.6 Web Services

For purposes of management-level system integration, a common, preferably open format for access to BAS data points (and possibly management functions as well) that is aligned with IT standards is a key asset. HTTP and HTML certainly are popular in BAS as well, but are limited to user interfaces since HTML is ill-suited for machine-to-machine communication. Therefore, by tradition, OPC [23] has fulfilled this role. However, it is in its classic design incarnation restricted to Windows platforms due to its use of OLE/DCOM (Object Linking and Embedding/Distributed Component Object Model).

Today, solutions emerge that follow the current mainstream trend of using WS for machine-to-machine interaction over a network. WS are entirely platform independent. They can be implemented using any programming language and run on any hardware platform or operating system. This maximizes interoperability and greatly improves the reusability of the services they provide. Central to WS are the use of XML for data representation and a standardized resource access protocol (typically SOAP, sometimes also plain HTTP). WS follow a modular concept. This allows the use of off-the-shelf standards for transmission, eventing, discovery, security, and many more. However, a drawback is the additional overhead introduced by the XML encoding (especially if small amounts of data are transmitted frequently) and the fact that servers cannot push information to clients (unless the latter implement a server themselves). Given the hardware resources available at the management level, this is typically not an issue, though.

The commitment to use WS technology determines low-level aspects of data representation and transmission. Still, the application domain remains to be modeled. Today, three standards hold particular promises for bringing the world of WS to BA. First, an OPC XML variant for access to data points has been defined, and implementations are available today. Moreover, the forthcoming OPC UA (Unified Architecture) will, besides other significant architectural improvements, also be centered on XML and WS (but has not yet been disclosed to the public at large). Second, BACnet was extended with a WS interface (BACnet/WS). Third, the oBIX (Open Building Information Exchange) initiative added another challenger. All three provide historical data access and event and alarm management besides a data point abstraction. Both the BACnet/WS and oBIX standards, which have been developed with special regard to building automation, are freely available to the general public.

29.5.6.1 BACnet/WS

Although published as BACnet Addendum 135-2004c in 2006, BACnet/WS (Building Automation and Control Network–WS) [24] is not tied to BACnet as an underlying BAS. It can equally be used to expose data from non-BACnet systems—although a mapping between the BACnet and BACnet/WS data models is, of course, defined.

The fundamental primitive data element in the BACnet/WS data model is the *node*. Nodes are arranged in a tree structure, having one mandatory parent node (except the root node) and an arbitrary number of children. From the network point of view, a node is mainly a static collection of *attributes*. An attribute represents a single aspect or quality of a node, such as its value or writability. Each attribute has a value of a specific type. A static list of primitive value types is defined that includes well-known types such as Boolean, Integer, Real, String and more, but also an enumeration type (called Multistate). Besides, an attribute may also be an array of a primitive value type. For some attributes, BACnet/WS supports localization (also for multiple locales simultaneously), for example, a temperature value may be available in Fahrenheit, Celsius, and Kelvin at the same time, or a name in various languages. Attributes may be mandatory or present under specific conditions only. They may also have subattributes.

Nodes and attributes are identified by their position in the tree structure. The hierarchical structure of nodes is reflected as a hierarchy of identifiers, resulting in a path used for naming. The

node identifiers are delimited by a slash, with the attribute identifier following after a colon (e.g., `/building_1/west_wing/boiler/temp:value`).

Each node has a mandatory attribute specifying the node's type. The node type may be chosen from a predefined list of strings. It describes the semantics of a node or subtree and determines the mandatory attributes. For example, a *point* node resembles a data point from the underlying system and has a mandatory *value* attribute. While a point is (reasonably) a leaf node, *area* nodes or *system* nodes are used as directory nodes and provide semantic information for their subtree (e.g., "west wing" or "HVAC"). The special node type *reference* allows multiple use of a node's data in the hierarchy by providing the path to its referent and reflecting most of its attributes.

BACnet/WS operate upon node attributes. Services are provided for retrieving and modifying single primitive values as well as "plural" versions of these services operating on a collection of arbitrary multiple nodes in order to reduce network traffic by batching the operations. Additional services allow read-only access to entire arrays, historic data, and locale information.

29.5.6.2 oBIX

The oBIX initiative is hosted by OASIS [25]. The oBIX specification defines a flexible object model for describing the data and operations available on a server. In oBIX, everything is an object: objects are also used to describe data types (classes) and operations (method signatures). Together with the possibility to custom define any kind of object by subtyping and composition, this makes the oBIX data model highly extensible.

Conceptually, every object is derived from the *root object*. It specifies a number of mandatory attributes, such as an object's name, which are inherited by all other objects. Like in every data model, various object types holding primitive values are defined. These are Booleans, Integer and Floating point numbers, enumerations, Strings, points in time and time spans. oBIX base object types also cover a number of universally applicable concepts: lists and feeds (containers with either static content or event queue semantics), errors, references to other objects, and operations. Custom classes can be derived from any object type. Multiple inheritance is supported. Every custom-built class can be handled by any oBIX client.

For flagging primitive values as coming from the automation system, oBIX offers Point classes. Read-only points are effectively an empty class (acting merely as a semantic marker); writable points additionally specify an operation for altering the corresponding value. oBIX also provides a very flexible SI-based system for describing engineering units, which is again based on objects.

Furthermore, the oBIX standard library contains a couple of predefined special purpose classes which a client can use to access functions of an oBIX server. This includes a service which relieves the client from monitoring an object's state (Watches), batch operations to reduce communication overhead, and historical data access.

Historical trends can be represented via the oBIX History Record, which groups a point value and a time stamp. The History object consists of a list of history records and methods to query them. Query filters can be specified and extended to a rollup calculation (e.g., for average values).

Eventually, oBIX defines a normalized model to query, watch, and acknowledge alarms. An oBIX server supplies feeds of alarm objects. Every time the server detects that the value of an object meets a predefined alarm condition, an alarm object is added to the feed. This object contains a timestamp and points to its source. Alarms can be stateful (i.e., the point in time when the source returned to normal is recorded) and they can record if (and by whom) the alarm was acknowledged.

oBIX follows what is called a RESTful approach (Representational State Transfer), a resource centric architectural style for WS. Central concepts of the RESTful approach are resources that share a uniform interface and a highly restricted set of operations on these resources. This approach mimics the World Wide Web where only the commands GET, PUT, and POST are used to access countless resources.

The same is true for oBIX services: only three network request types are defined at the WS level. The first two of them are *read* (applicable to any object) and *write* (for writable objects). For operations beyond basic "get" and "set," custom operations can be defined on the oBIX level, just as any other custom object. For invoking these operations a third base network request type *invoke* is provided.

Naming in oBIX is realized via two different, complementary concepts. First, every object can have a *name*. It is used to identify a subobject within a composite object (e.g., consider an object with two string members or two operations—their name tells them from each other). Second, every object can be assigned a URI (an *href*). This is necessary whenever an object is to be referenced from the outside. To apply any network request (read, write, or invoke) to an object, this reference is required. No higher-level semantics are associated with the URI namespace.

29.5.6.3 Comparison

The main difference between BACnet/WS and oBIX lies in their data model. While BACnet/WS provides a rather static type system and arranges nodes in a tree hierarchy, the oBIX data model is far more flexible and does not predefine a particular arrangement of application related objects. For example, the oBIX unit system is based on the flexible combination of SI base units, while BACnet/WS provides a predefined collection of text strings. As another, BACnet/WS offers a fixed set of services for attribute access, while oBIX allows adding custom operations. While the simplicity of the BACnet/WS data model makes it easily understandable, oBIX provides more possibilities due to its extensibility.

In addition to SOAP, oBIX also supports a plain HTTP protocol binding. Both oBIX and BACnet/WS leave security to the transmission channel and recommend the use of standard WS security like HTTP over SSL/TLS and the WS-Security standard. oBIX further suggests implementing permission-based degradation, which basically means that users should only see (and thus manipulate) the objects they have permission for.

29.5.7 Home Automation

Adopting automation technology is equally attractive in the residential domain. Besides efficiency considerations, increased comfort and peace of mind are key motives here. Also, elderly people can live in their own homes longer (ambient assisted living, smart home care). The importance of this aspect continuously increases as life expectancy does.

Despite having a considerable number of things in common, there is a difference between home automation (residential settings) and building automation (functional buildings). Since mechanical ventilation is seldom present in residential settings (although continuing attention to low energy consumption can be expected to change this situation in the long run), lighting and shading control have a more visible position in home automation than HVAC. Considerable emphasis is placed on the integration of consumer electronics (entertainment systems) and household appliances.

Building automation focuses on the goals of energy efficiency, optimized management, and detailed accounting. While energy efficiency is a growing concern in the residential domain as well, the additional comfort and peace of mind a home automation system promises are still influential decision factors. One obvious difference is the scale of the system: The number of devices involved in a home automation project is by orders of magnitude lower than for large building automation projects.

Nonetheless, the complexity of home automation projects must not be underestimated, not at least since a large number of application domains must be integrated. All field, automation and management functions mentioned are relevant. Both equipment as well as commissioning and maintenance cost have to be kept very low. Easy configuration is of particular importance, since the disproportionately high setup cost will otherwise reduce the attractiveness of automation. This favors

robust "plug-and-play" systems with the capability of communicating via the mains or wireless. Acknowledging these specific challenges, one explicitly speaks of "home automation" or "domotics." The term "building automation" will remain reserved for functional buildings.

A large number of proprietary, mostly centralized solutions is available. Many of them are application domain specific, a certain focus on multimedia aspects can be noted. Published networking and middleware standards of particular relevance are X10 for simple control via PL (in North America), EHS/KNX [16] for household appliances* and KNX RF (in Europe), ECHONET (in Japan), and UPnP (mainly for multimedia applications). DPWS (Device Profile for WS) is intended as a WS conformant successor to UPnP. Regarding unpublished consortium standards, the OpenTherm protocol has reached importance in Europe for point-to-point communication in home heating systems, while Z-Wave has gained momentum in North America for wireless automation purposes. Dynamic application frameworks—OSGi [26] being the prime, open example—allow providing residential gateways with the flexibility required to have multiple service providers connect from remote to a variety of devices within the home environment. Remote access can be expected to increase in importance as building services equipment, alarm systems, and entertainment systems in homes increase in features and complexity.

29.5.8 Standardization

A communication protocol typically is designed by a single company. If this company decides that it may profit from others using this protocol, it may choose to promote it itself or hand it over to a company consortium or user group. Publication of such a company or consortium standard as a formal standard by an official standards body is largely considered the apex of recognization and stability. Hence, adherence of equipment to such formal standards is required in an increasing number of tenders.

Officially recognized standards bodies ensure that the standards they maintain and publish fulfill the conditions of open systems as outlined, i.e., nondiscriminatory access to specification and licensing. In addition, they follow an open decision process with the goal of allowing everyone interested in the development of a standard to be heard. Likewise, evolution of standards is consensus controlled. Within the purview of an official standards body, there should theoretically be a single formal standard for any particular design or procedure to be standardized. However, work areas of committes and working groups overlap, and some of them have also elected to publish standards that include several implementation variants. In the following, committees and working groups of European and international standards bodies concerned with BAS communication protocols are presented as examples.

ISO TC 205† (Building Environment Design) is publishing a series of international standards (ISO 16484) under the general title of Building Automation and Control Systems (BACS). The series includes a generic system model describing hardware [6] and functions [4] of a BACS. It also contains the BACnet standard [10].

CEN‡ TC 247 (Building Automation, Controls, and Building Management) was responsible for paving the way in European BA protocol standardization through cumulative prestandards of industry-standard protocols for the automation and field level (ENV 13154-2 and ENV 13321-1), which also included a collection of standardized object types for the field level (ENV 13154-1). TC 247 also

* CECED (Conseil Européen de la Construction d'Appareils Domestiques, European Committee of Domestic Equipment Manufacturers) has defined device profiles for home appliances under the name of CHAIN (CECED Home Appliances Interoperating Network), but has not made them public to date.

† International Standards Organization, Technical Committee 205.

‡ Comité Européen de Normalisation, European Committee for Standardization.

made significant contributions to ISO 16484. More recently, TC 247 has published the KNXnet/IP (EN 13321-2) and LonWorks (EN 14908) communication protocols.

CENELEC* TC 205 (HBES) oversees the EN 50090 series, a standard for all aspects of HBES tightly coupled to KNX. In addition, it considers matters of EMC (electromagnetic compatibility) and electrical and functional safety.

ISO/IEC JTC1 SC25 WG1† (Information Technology, Home Electronic System) focuses on the standardization of control communication within homes. WG1 has elected to include various communication protocols in its ISO/IEC 14543 series, starting with KNX, but with the intent for others such as LonTalk and ECHONET to follow. In addition, it is working on interoperability guidelines (ISO/IEC 18012) and a residential gateway (ISO/IEC 15045). Despite its focus on the home environment, the scope of WG1 does not exclude commercial buildings. This especially concerns its interoperability guidelines [27] when considering field level functions.

29.6 Conclusion and Outlook

Building automation can increase comfort and decrease energy consumption at the same time. A trend towards further decentralization of intelligence and flexible allocation of system functionality to devices can be seen to emerge. Open communication standards rise in importance.

While the individual levels (field, automation, management) were clearly associated with physical devices in the past, these devices now often implement functions belonging to multiple levels. Also, these functions are more flexibly distributed today. Intelligent field devices are not the only example; for instance, it would equally be possible to place trend server functionality in routers in a distributed way.

The intelligent building of the future cannot be realized without increased convergence and fusion of IT and BAS. However, this is not feasible if individual data points are taken as the starting point due to the excessive effort required. Thus, domain-specific functions (e.g., HVAC control) at first and, in the following, complex global functions involving multiple trades and application domains (e.g., energy management, or building performance monitoring) must be defined and standardized. Efforts in this direction are undertaken, among others, by the oBIX community.

Today, hardly any of the data gathered during the construction phase (in particular, information about the building structure) are reused in BAS, although they are already present in machine readable form—often even using standardized specification frameworks, such as the Industry Foundation Classes [28]. Tapping this resource for BAS engineering and operation purposes certainly would be of benefit.

Another aim that needs to be pursued in parallel is to not only integrate systems that currently operate stand-alone at the management level, but also bring them together on a single, common control network.

For example, the state of doors and windows is of importance to both the HVAC system (to avoid heating or cooling leakage to the outdoor environment) and the security system (to ensure proper intrusion protection at night). Also, motion detectors can provide intrusion detection at night and automatic control of lights during business hours. Such common use of sensors in multiple control domains can reduce investment and operational cost. Control information can also be derived from CCTV imagery through image processing techniques. For example, presence detection and people counting for better HVAC or elevator control can be achieved this way. As another benefit, the state

* Comité Européen de Normalisation Electrotechnique, European Committee for Electrotechnical Standardization.
† ISO/International Electrotechnical Commission Joint Technical Committee 1, Subcommittee 25, Working Group 1.

of doors and windows can be detected. Data from multiple sensors may also be fused for additional expressiveness.

Such integration creates special requirements on the underlying control network. Since the requirements of safety and security applications differ from those of HVAC control applications, strict "vertical" separation into systems serving functional domains traditionally persists in building automation network and system architectures.

The SafetyLon project has led the way in this regard by defining a framework for IEC 61508 compliant communication via LonWorks. SafetyLon enables devices implementing safety relevant applications to coexist on a shared medium with standard LonWorks devices.

Generally speaking, however, integration at the device level introduces a level of complexity that still remains a challenge to be handled. As another particular issue, building automation protocols in use today offer hardly any security support, which is a significant problem when security functions are considered for integration. Of the open systems discussed, a simple authentication procedure can be found in LonWorks, while KNX provides no security mechanisms whatsoever. The BACnet security infrastructure is well developed in comparison, but also has certain weaknesses [29].

The level of communications security provided by the variety of proprietary, undocumented protocols mostly proved to be appropriate for isolated BAS. Nowadays security concerns are increasing rapidly, however. For example, remote access is standard on present-day systems. Protection against denial-of-service attacks becomes more of an issue as buildings get more dependent on automation systems. Open media such as PL and, especially, radio communication further increase vulnerability, since access to the medium can be gained in an unobtrusive manner. Further, the shift to open systems reduces the knowledge barrier for intruders.

Any discussion regarding BAS and their implementation is not complete without addressing design issues. Today's systems are still being installed without formal specification or design. System integrators have to rely on perceived wisdom, experience, and best practice. Thus, techniques for better prediction of performance and dependability are required along with automated tools to support this. Up to now, despite the existence of sophisticated management tools, complete analysis or precise modeling of the distributed application is still beyond reach. These problems need to be addressed and leave ample place for future research activities.

References

1. K. Daniels, *Advanced Building Systems: A Technical Guide for Architects and Engineers*, Birkhäuser, Basel, 2003.
2. J. K. W. Wong, H. Li, and S. W. Wang, Intelligent building research: A review, *Automation in Construction*, 14(1), 143–159, 2005.
3. C. P. Underwood, *HVAC Control Systems: Modelling, Analysis and Design*, Routledge, New York, 1999.
4. ISO 16484-3, *Building Automation and Control Systems (BACS)—Part 3: Functions*, 2005.
5. J. P. Thomesse, Fieldbuses and interoperability, *Control Engineering Practice*, 7(1), 81–94, 1999.
6. ISO 16484-2, *Building Automation and Control Systems (BACS)—Part 2: Hardware*, 2004.
7. IEC 60929, *AC-Supplied Electronic Ballasts for Tubular Fluorescent Lamps—Performance Requirements*, 2006.
8. EN 1434-3, *Heat Meters—Part 3: Data Exchange and Interfaces*, 1997.
9. ANSI/ASHRAE Std. 135, *BACnet—A Data Communication Protocol for Building Automation and Control Networks*, 1995–2004.
10. ISO 16484-5, *Building Automation and Control Systems (BACS)—Part 5: Data Communication Protocol*, 2007.
11. ANSI/EIA/CEA Std. 709.1, Rev. A, *Control Network Protocol Specification*, 1999.
12. EIA/CEA Std. 709.1, Rev. B, *Control Network Protocol Specification*, 2002.

13. EN 14908, *Open Data Communication in Building Automation, Controls and Building Management—Control Network Protocol*, 2005.
14. ANSI/EIA/CEA Std. 852, *Tunneling Component Network Protocols over Internet Protocol Channels*, 2002.
15. LonMark International, http://www.lonmark.org.
16. Konnex Association, *KNX Specifications, Version 1.1*, 2004.
17. EN 50090, *Home and Building Electronic Systems* (*HBES*), 1994–2007.
18. ISO/IEC 14543, *Information Technology—Home Electronic System* (*HES*) *Architecture*, 2006–2007.
19. EN 13321-2, *Open Data Communication in Building Automation, Controls and Building Management—Home and Building Electronic Systems—Part 2: KNXnet/IP Communication*, 2005.
20. RFC 4944, *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*, 2007.
21. IEEE Std. 802.15.4, *Wireless Medium Access Control* (*MAC*) *and Physical Layer* (*PHY*) *Specifications for Low-Rate Wireless Personal Area Networks* (*WPANs*), 2003–2007.
22. ZigBee Alliance, http://www.zigbee.org.
23. OPC Foundation, http://www.opcfoundation.org.
24. American Society of Heating, Refrigerating and Air-Conditioning Engineers, *ANSI/ASHRAE Addendum C to ANSI/ASHRAE Standard 135-2004*, 2006.
25. OASIS, *oBIX 1.0 Committee Specification 01*, 2006.
26. OSGi Alliance, *OSGi Service Platform, Release 4, Version 4.1*, 2007.
27. ISO/IEC 18012-1, *Information Technology—Home Electronic System—Guidelines for Product Interoperability—Part 1: Introduction*, 2004.
28. International Alliance for Interoperability, http://www.iai-international.org.
29. W. Granzer, W. Kastner, G. Neugschwandtner, F. Praus, Security in networked building automation systems, *Proceedings of the 6th IEEE International Workshop on Factory Communication Systems*, Turin, June 28–30, pp. 283–292, 2006.
30. W. Kastner, G. Neugschwandtner, S. Soucek, and H. M. Newman, Communication systems for building automation and control, *Proceedings of the IEEE*, 93(6) 1178–1203, 2005.

Contributor Index

Aarabi, Parham	Distributed Signal Processing in Sensor Networks, 9-1–9-20
Behnke, Ralf	Developing and Testing of Software for Wireless Sensor Networks, 12-1–12-36
Beutel, Jan	Wireless Sensor Networks Testing and Validation, 11-1–11-27
Blumenthal, Jan	Resource-Aware Localization in Sensor Networks, 6-1–6-27
	Developing and Testing of Software for Wireless Sensor Networks, 12-1–12-36
Cena, Gianluca	Controller Area Networks for Embedded Systems, 15-1–15-38
	Hybrid Wired/Wireless Real-Time Industrial Networks, 26-1–26-20
Chatterjea, S.	Architectures for Wireless Sensor Networks, 4-1–4-33
Collotta, Mario	Energy-Efficient MAC Protocols for Wireless Sensor Networks, 8-1–8-24
Dulman, Stefan	Introduction to Wireless Sensor Networks, 3-1–3-11
	Architectures for Wireless Sensor Networks, 4-1–4-33
Dzung, Dacfey	Design and Implementation of a Truly-Wireless Real-Time Sensor/Actuator Interface for Discrete Manufacturing Automation, 28-1–28-28
Elmenreich, Wilfried	Configuration and Management of Networked Embedded Devices, 22-1–22-21
Endresen, Jan	Design and Implementation of a Truly-Wireless Real-Time Sensor/Actuator Interface for Discrete Manufacturing Automation, 28-1–28-28
Felser, Max	Real-Time Ethernet for Automation Applications, 21-1–21-20
Frey, Jan-Erik	Wireless Sensor Networks for Automation 27-1–27-43
	Design and Implementation of a Truly-Wireless Real-Time Sensor/Actuator Interface for Discrete Manufacturing Automation, 28-1–28-28
Galla, Thomas M.	Standardized System Software for Automotive Applications, 18-1–18-29
Gill, Christopher D.	Middleware Design and Implementation for Networked Embedded Systems, 2-1–2-18
Golatowski, Frank	Developing and Testing of Software for Wireless Sensor Networks, 12-1–12-36
Havinga, Paul J. M.	Introduction to Wireless Sensor Networks, 3-1–3-11
	Architectures for Wireless Sensor Networks, 4-1–4-33
Jahromi, Omid S.	Distributed Signal Processing in Sensor Networks, 9-1–9-20
Kastner, Wolfgang	Data Communications for Distributed Building Automation, 29-1–29-34
Lennvall, Tomas	Wireless Sensor Networks for Automation 27-1–27-43
Lo Bello, Lucia	Power-Efficient Routing in Wireless Sensor Networks, 7-1–7-40
	Energy-Efficient MAC Protocols for Wireless Sensor Networks, 8-1–8-24
Matheus, Kirsten	Wireless Local and Wireless Personal Area Network Communication in Industrial Environments, 25-1–25-23

Millinger, Dietmar	FlexRay Communication Technology, 16-1–16-16
Moyne, James R.	Networked Control Systems for Manufacturing: Parameterization, Differentiation, Evaluation, and Application, 23-1–23-37
Navet, Nicolas	Trends in Automotive Communication Systems, 13-1–13-24
Neugschwandtner, Georg	Data Communications for Distributed Building Automation, 29-1–29-34
Nossal-Tueyeni, Roman	FlexRay Communication Technology, 16-1–16-16
Obermaisser, Roman	Time-Triggered Communication, 14-1–14-16
Prüter, Steffen	Developing and Testing of Software for Wireless Sensor Networks, 12-1–12-36
Rajnak, Antal	LIN Standard, 17-1–17-14
	Volcano: Enabling Correctness by Design, 19-1–19-20
Reichenbach, Frank	Resource-Aware Localization in Sensor Networks, 6-1–6-27
Sauter, Thilo	Fieldbus Systems: Embedded Networks for Automation, 20-1–20-64
Schaefer, Guenter	Sensor Network Security, 10-1–10-30
Scheible, Guntram	Design and Implementation of a Truly-Wireless Real-Time Sensor/Actuator Interface for Discrete Manufacturing Automation, 28-1–28-28
Simonot-Lion, Françoise	Trends in Automotive Communication Systems, 13-1–13-24
Su, Weilian	Overview of Time Synchronization Issues in Sensor Networks, 5-1–5-12
Subramonian, Venkita	Middleware Design and Implementation for Networked Embedded Systems, 2-1–2-18
Thiele, Lothar	Wireless Sensor Networks Testing and Validation, 11-1–11-27
Tilbury, Dawn M.	Networked Control Systems for Manufacturing: Parameterization, Differentiation, Evaluation, and Application, 23-1–23-37
Timmermann, Dirk	Resource-Aware Localization in Sensor Networks, 6-1–6-27
	Developing and Testing of Software for Wireless Sensor Networks, 12-1–12-36
Toscano, Emanuele	Power-Efficient Routing in Wireless Sensor Networks, 7-1–7-40
	Energy-Efficient MAC Protocols for Wireless Sensor Networks, 8-1–8-24
Valenzano, Adriano	Controller Area Networks for Embedded Systems, 15-1–15-38
	Hybrid Wired/Wireless Real-Time Industrial Networks, 26-1–26-20
Vitturi, Stefano	Hybrid Wired/Wireless Real-Time Industrial Networks, 26-1–26-20
Willig, Andreas	Wireless LAN Technology for the Factory Floor: Challenges and Approaches, 24-1–24-20
Woehrle, Matthias	Wireless Sensor Networks Testing and Validation, 11-1–11-27
Zurawski, Richard	Networked Embedded Systems: An Overview, 1-1–1-16

Subject Index

A

ABS, *see* Anti-lock braking system
ACA, *see* Autonomous component architecture
Acquisitional query processing, 4-24, 4-28
Active damage detection, ping node scheduling, 2-3–2-4
ADAPTIVE Communication Environment (ACE), 2-6
Adaptive, Information-centric, and Lightweight MAC (AI-LMAC) protocol, 4-29
Adaptive threshold-sensitive energy-efficient sensor network (APTEEN) protocol
  queries, 7-23
  super-frame comparisons between LEACH and MECH, 7-24
Ad hoc networks, 10-3, 24-6, 24-15, 29-9–29-10
Ad hoc on-demand distance vector routing (AODV), 12-31
ADV messages, 7-7–7-8
Aggregation operation, interval-based communication scheduling, 4-26–4-27
Aircraft fieldbusses, automotive, 20-57
Analog-to-digital converter models, 12-23
Anti-lock braking system (ABS), 13-1
AODV, *see* Ad hoc on-demand distance vector routing
Application layer, 4-5, 4-8
Application objects (AOs), 29-27
Application programming interface (API), 18-6, 18-8
APTEEN protocol, *see* Adaptive threshold-sensitive energy-efficient sensor network protocol
Association for Standardization of Automation and Measuring Systems (ASAM), 16-15, 18-12
  flash programming, 18-17
  online calibration feature, 18-14–18-15
  protocol stack, 18-12
  synchronous data transfer feature, 18-14
  XCP packet, structure of, 18-13
  XCP protocol layer, 18-13
  XCP transport layers, 18-13–18-14
Assumption-based coordinates algorithm, 6-23–6-24
Asymmetric delay, 5-3
Authenticated broadcast, 10-12
Automatic repeat request (ARQ) scheme, 24-11–24-12
Automation pyramid, hierarchical network levels, 20-5
Automotive communication systems
  car domains, 13-2–13-3
  communication system services, 13-21
  different networks, requirements, 13-3–13-4
  event-triggered *vs.* TT, 13-4
  multiplexed communications, point-to-point, 13-1
  optimized networking architectures, 13-20
Automotive electronic systems, design, 13-14
Automotive industry
  needs, 17-2
  software, 19-3
Automotive networked embedded systems
  cooperative development process, 1-7
  FlexRay protocol, 1-5–1-6
  functional domains for deployment, 1-4
  TTP/C protocol, 1-4–1-5
Automotive open system architecture, 16-16
Automotive systems, hardware architecture, 18-2
Autonomous component architecture (ACA), 12-30
Automotive Open System Architecture (AUTOSAR)
  application software architecture, 18-16
  consortium, 13-17
  JasPar, relationship, 18-27
  OEM manufacturers, 18-15
  standard
    definition, 13-18
    MW interface, 13-19

sender–receiver communication model, 13-17
system software architecture
communication substack, 18-19–18-21
I/O substack, 18-22–18-23
memory substack, 18-21–18-22
system services substack, 18-23–18-26
system software modules, 18-16–18-19
runnable entities, 18-16

B

BACnet, *see* Building automation and control network
Bandwidth, 12-12, 17-8
Bandwidth, industrial network
definition, 23-4
frame time, 23-5
message transmission time, 23-5
Base station-controlled dynamic clustering protocol (BCDCP)
key points, 7-26
network functioning, 7-27
BAS, *see* Building automation system
Berkeley MAC protocol (B-MAC protocol)
interfaces, 8-9
main components, 8-8–8-9
BG, *see* Bus guardian
Binary exponential backoff (BEB) algorithm, 23-15–23-16
Bit stuffing technique, 15-6
Bluetooth technology (BT)
advantages, 23-22
asynchronous connection-less (ACL), 25-6
determinism characterization, 23-22–23-24
drawbacks, 23-22
factory floor units, 25-8
forward error correction (FEC), 25-5
frequency hopping (FH), 25-4–25-5
machine-to-machine communication, 25-7
master–slave concept, 25-5
physical layer, 25-5
piconet and slaves, 25-5–25-6
sophisticated approach, 25-8
synchronous connection oriented (SCO), 25-6–25-7
triple transmission rates, 25-7
B-MAC protocol, *see* Berkeley MAC protocol
Bosch GmbH, 15-1
Bottom-up approach, 2-6
Bounding box algorithm, 6-20–6-21
Breadth first spanning tree, 10-6–10-7
Building automation and control network (BACnet)
application layer data unit (APDU), 29-19
BACnet interoperability building block (BIBB), 29-20
binary output object, 29-19
degree of freedom, 29-20
design and description, 29-18
segments, 29-18–29-19
Building automation system (BAS)
automation level, 1-12
benefits
automatic feedback control, 29-3
building management system (BMS), 29-4
control, 29-5–29-6
lighting systems demand control, 29-3
system integration, 29-4–29-5
communication architecture supporting, 1-12
communication characteristics and strategies
carrier sense multiple access (CSMA), 29-11
distributed control approach, 29-10
I/O functions, 29-10
network protocols, 29-11
data communication services
ad hoc operation, 29-9–29-10
semantics, 29-8
shared-variable model, 29-9
data flow, 29-6–29-7
data points, 29-6
functions and services distribution
dedicated special systems (DSS), 29-12
direct digital control (DDC) stations, 29-12
field, automation, and management level, 29-16
gateways, 29-15
intelligent field device, 29-12
IP technology, 29-15
layer model, 29-12–29-13
network backbone, 29-14
tunneling router, 29-15
input/output (I/O) and management functions, 29-6
network nodes, 1-12–1-13
Neuron C programs, 1-13
objective, 1-11
open standards
BACnet/Web serive (WS), 29-28–29-29
building automation and control network (BACnet), 29-18–29-20
home automation, 29-30–29-31
IEEE 802.15.4, 29-26–29-27
KNX, 29-22–29-25
LonWorks, 29-20–29-22
open building information exchange (oBIX), 29-29–29-30
plant room and field level subsystems, 29-17–29-18
standardization, 29-31–29-32
ZigBee, 29-27
processing functions, 29-6
services, 1-11–1-12, 29-2–29-3
Bus guardian (BG), 16-10–16-11

C

Cached connection strategy, 2-11
CADR protocol, *see* Constrained anisotropic diffusion routing protocol
CAMAC, *see* Computer automated measurement and control
CAN 2.0A data frame format, 13-6
CAN application layer (CAL), 15-3
CAN-based safety bus, 20-56
CAN in Automation (CiA), 15-3
CANopen device, 20-8

conceptual internal architecture, 15-26
device behavior
digital input devices, 15-36–15-37
digital output devices, 15-37
device definition, 15-34
default PDOs, 15-36
profile predefinitions, 15-35–15-36
CANopen network, 15-36
CAN, *see* Controller area network
Cardbus interface, 14-12
Carrier sense multiple access (CSMA), 15-2, 15-9, 29-11
Centroid localization (CL), 6-16
CiA, *see* CAN in Automation
CIM, *see* Computer-integrated manufacturing
Cipher block chaining MAC (CBC-MAC) construction, 10-10
Clear channel assessment (CCA), 8-9
Clock glitches, 5-3
Clock synchronization, 16-7
Cluster-based routing protocols
APTEEN protocol, 7-23–7-24
BCDPC protocol, 7-26–7-27
energy-aware routing protocol, 7-27–7-28
energy saving, 7-6
extension of LEACH protocol, 7-19–7-20
HEED protocol, 7-20–7-21
LEACH-C protocol, 7-20
LEACH protocol, 7-17–7-19
MECH protocol, 7-21–7-22
PEGASIS protocol, 7-24–7-26
TEEN protocol, 7-22–7-23
Clustered approach to in-network processing, 4-26
CMD, *see* Command packets
CNI, *see* Communication network interface
Coarse-grained localization, 6-16–6-17
Co-channel interference, 24-7
Command packets (CMD), 18-13
Command transfer object, 18-13
Communication
controllers, 18-3
interoperability, 20-42
network
fieldbus systems, 20-2
interface, 4-3, 14-2
SO/IEC 8802-3, 20-53
profiles, fieldbus user groups, 20-41
scheduling, 16-14
software components, 13-18
subsystem implementation
antenna diversity and switching, 28-8
base station, 28-9–28-10
Bluetooth-equipped laptop, 28-13, 28-15, 28-17
frequency hopping, 28-6, 28-8–28-9
frequency usage, 28-12
IEEE 802.15.4/Zigbee/WirelessHART, 28-13
industrial electromagnetic interference, 28-14–28-15
laptop PC, 28-15–28-16
medium access and retransmission, 28-6–28-7
multicell operation, 28-8–28-9
performance figure, 28-12
performance measurements, 28-13–28-14
robustness, 28-11–28-12
sensor module, 28-10
spectral efficiency, 28-12
time, frequency and location, 28-10–28-11
WLAN, 28-12–28-13
technologies
publisher–subscriber model, 12-39
sensing and actuator devices, 12-1
Communication profile families (CPF), 21-3
Computer automated measurement and control (CAMAC), 20-7
Computer-integrated manufacturing (CIM), 20-4
Concealed data aggregation scheme, 10-26
Conceptual architecture
digital inputs, 15-36
digital outputs, 15-37
Configuration and management
applications, 22-1–22-2
fieldbus system, 22-2
interfaces
architecture, 22-6
interface file system (IFS) approach, 22-5–22-7
plug and play *vs.* plug and participate, 22-3
profiles, datasheets, and descriptions
electronic device description language (EDDL), 22-8
field device tool/device type manager (FDT/DTM), 22-9
interface file system/smart transducer descriptions, 22-10–22-11
transducer electronic datasheet (TEDS), 22-9–22-10
requirements, 22-3–22-5
smart or intelligent devices, 22-3
Constrained anisotropic diffusion routing (CADR) protocol, 7-12
Contiki
event-driven kernel, 12-18
interprocess operations, 12-20
libraries, 12-19–12-20
memory footprint, 12-20
runtime programming, 12-20
system architecture, 12-19
TCP/IP stack, 12-18
Controller area network (CAN), 13-2, 15-1, 24-11, 24-13
advantages, 15-18–15-19
arbitration phase, 13-7, 15-10
automatic retransmission feature, 15-22
automotive applications, 15-2–15-3
blocking time and interference, 15-17
CANopen
network management, 15-25–15-26
object dictionary, 15-26–15-27
real-time process data, 15-27–15-30
service data objects, 15-30
data link layer, 14-10

digital output devices, 15-37
drawbacks
dependability, 15-20–15-21
determinism, 15-20
performance, 15-19–15-20
ECU, 19-3
electrical interface, node to bus, 15-5
EMCY object, 15-31
hardware, 19-4
HIS CAN driver, 18-30
ID, predefined connection set, 15-34
industry-standard solution, 19-3
network interface, 19-7
network management
error control services, 15-33
node control services, 15-31–15-33
predefined connection set, 15-33–15-34
problems, 19-3
protocol architecture
communication services, 15-12–15-13
data-link layer, 15-4
error management, 15-10–15-11
fault confinement, 15-11–15-12
frame format, 15-6–15-8
implementation, 15-13–15-14
ISO OSI model, 15-3
medium access technique, 15-9–15-10
network topology, 15-4–15-5, 15-5–15-6
physical layer, 15-4
protocol stack, 15-4
schedulability analysis, 15-16–15-18
communication requirements, 15-14–15-16
networked embedded system, usage, 15-18
software layers, 18-9
solutions, 15-21
standard, data link layer, 13-8
SYNC producer, 15-30–15-31
time-triggered
implementation, 15-23–15-24
main features, 15-21–15-22
protocol specification, 15-22–15-23
transmission delay, 15-17
transmission errors, 15-11
worst-case response time, 15-16
Control network protocol (CNP), 29-20–29-21
Convex feasibility approach, 9-3
projection algorithms, 9-18–9-19
Convex position estimation (CPE), 6-22–6-23
CORBA standard
faithful implementation, 2-6
object adapters supported by ORB, 2-9
Correlation-based collaborative MAC protocol
aim, 8-14
E-MAC and N-MAC, 8-14–8-15
evaluation, 8-15–8-16
Cougar protocol, 7-12
CPE, *see* Convex position estimation
Crankshaft protocol, 8-13–8-14
CRC, *see* Cyclic redundancy check; Cyclic redundancy code
Cross-domain integration, 29-4
Cross-layer optimization, 4-24–4-25
CSMA/CD protocol, 23-15–23-16
CSMA-collision avoidance (CSMA-CA) scheme, 29-26–29-27
CSMA, *see* Carrier sense multiple access
Cyclic redundancy check (CRC), 15-6
Cyclic redundancy code (CRC), 14-7

D

DALI, *see* Digital addressable lighting interface
Data acquisition (DAQ) list, 18-14
Data-centric data/query dissemination, 4-24–4-25
Data-centric routing protocols, 4-8
Cougar approach, 7-12
data aggregation, 7-5
directed diffusion mechanism, 7-8–7-10
IDSQ/CADR, 7-12
rumor routing, 7-10–7-11
SPIN, 7-6–7-8
TTDD protocol, 7-13
Data delivery model, 7-3
Data encoding scheme, 4-6
Data frames format, 15-7
Data-gathering MAC protocol (DMAC protocol), 8-8
Data length code (DLC), 15-7
Data-link layer, 4-5, 4-7–4-8
DATA messages, 7-7–7-8
Data transfer object, 18-13
Data transmission, 20-10
DCF protocol, *see* Distributed coordination function (DCF) protocol
Dcm, *see* Diagnostic communication manager
DECOMSYS OEM-supplier development process, 16-15
DECT, *see* Digital European cordless telephone
Dedicated special systems (DSS), 29-12
Dem, *see* Diagnostic event manager
Denial of service (DoS) attacks
breaking into sensor nodes, 10-4
countermeasures against, 10-5–10-6, 10-7
flooding, 10-6
forwarding and routing, 10-5–10-6
jamming, 10-5
Deployment-support network, 11-9–11-10
DeviceNet network, 15-3
CAN-based network, 23-8, 23-14
device delays, request–response setup, 23-6–23-7
message frame format, 23-14
packet interval, 23-24
strobed message connection, 23-9
Diagnostic communication manager (Dcm), 18-21
Diagnostic event manager (Dem), 18-25
Diagnostic management, 18-11
Diagnostic trouble codes (DTCs), 18-25
Diesel particulate filter (DPF), 13-20
Digital addressable lighting interface (DALI), 29-17–29-18
Digital European cordless telephone (DECT), 24-15
Direct digital control (DDC) stations, 29-12
Directed diffusion
publish/subscribe API, 4-30

basic idea, 7-8
energy efficiency, 7-10
functioning, 7-9
in-network processing, 4-26
reinforcement mechanism, 7-9
setting up of gradients, 4-30
vs. AODV, 7-10
Direct sequence spread spectrum (DSSS) mode, 24-8–24-9, 25-10–25-11
Disaster monitoring, wireless sensor networks, 3-4
Distance estimation methods
based on neighborhood information
connectivity, 6-7
hop count, 6-6–6-7
neighborhood intersection distance estimation scheme (NIDES), 6-9–6-10
three/two neighbor algorithm, 6-8–6-9
based on signal attenuation
minimal transmission power (MTP), 6-11–6-12
received signal strength (RSS), 6-10–6-11
based on time period
lighthouse localization system, 6-7–6-8
round trip time, 6-5–6-6
time difference of arrival (TDoA), 6-4–6-5
time of arrival (ToA), 6-2–6-4
Distributed and self-organizing scheduling algorithm (DOSA), 4-27
Distributed coordination function (DCF) protocol, 24-14–24-15
Distributed data extraction techniques
attributes of queries
duration, 4-18
result, 4-23
scope, 4-18, 4-23
essential conceptual building blocks in, 4-24–4-25
for WSNs
acquisitional query processing, 4-28
cross-layer optimization, 4-29
data-centric data/query dissemination, 4-30–4-31
in-network processing, 4-26–4-28
relationship between, 4-25
Distributed least squares (DLS) algorithm, 6-22
Distributed network of sensors, 9-1
DLC, *see* Data length code
DLS algorithm, *see* Distributed least squares (DLS) algorithm-
DMAC protocol, *see* Data-gathering MAC protocol
DOC middleware paradigms, 2-6
Domains, *see* Control network protocol (CNP)
DOSA, *see* Distributed and self-organizing scheduling algorithm
DoS attacks, *see* Denial of service (DoS) attacks
DPF, *see* Diesel particulate filter
DSS, *see* Dedicated special systems
DSSS mode, *see* Direct sequence spread spectrum (DSSS) mode
DTCs, *see* Diagnostic trouble codes

E

EAR protocol, *see* Energy-aware routing (EAR) protocol
EASIS, *see* Electronic Architecture and System Engineering for Integrated Safety Systems
ECU, *see* Electronic control unit
EDDL, *see* Electronic device description language
EIA-485-based protocol, 29-17
EIA-709, 1-12
EIB, *see* European installation bus
Electrical architectures, needs, 17-2
Electromagnetic compliance (EMC), 17-2
Electromagnetic interference (EMI), 13-4, 28-5, 28-14
Electronic Architecture and System Engineering for Integrated Safety Systems (EASIS), 18-2
Electronic control unit (ECU), 13-1, 18-2
architecture, 16-4
diode, 17-4
LIN specification, 17-4
NM module transits, 18-6
Electronic device description language (EDDL), 22-8
Electronic stability program (ESP), 13-1
Embedded control systems
CAN, automation, 15-3
controller area network, 15-1
EMC, *see* Electromagnetic compliance
EMI, *see* Electromagnetic interference
EmStar
components, 12-25
Crossbow Stargate devices, 12-25
EmCee, 12-26
IPC mechanism, 12-26
sample application, 12-25
sensor network visualization, 12-25
software environment, 12-24
UDP protocol, 12-26
Energy-aware QoS routing protocol, 7-31
Energy-aware routing (EAR) protocol, 7-27–7-28
data communication phase, 7-15–7-16
energy efficiency, 7-16
route maintenance phase, 7-16
setup phase, 7-15
Energy-saving routing protocols
cluster-based routing protocols
APTEEN protocol, 7-23–7-24
BCDPC protocol, 7-26–7-27
energy-aware routing protocol, 7-27–7-28
energy saving, 7-6
extension of LEACH protocol, 7-19–7-20
HEED protocol, 7-20–7-21
LEACH-C protocol, 7-20
LEACH protocol, 7-17–7-19
MECH protocol, 7-21–7-22
PEGASIS protocol, 7-24–7-26
TEEN protocol, 7-22–7-23
data-centric routing protocols
Cougar approach, 7-12
data aggregation, 7-5

directed diffusion mechanism, 7-8–7-10
IDSQ/CADR, 7-12
rumor routing, 7-10–7-11
SPIN, 7-6–7-8
TTDD protocol, 7-13
location-based routing protocols
GEAR protocol, 7-30–7-31
location information, 7-6
MECN protocol, 7-28–7-30
open issues concerning, 7-36
optimization-based routing protocols, 7-4
EAR protocol, 7-15–7-16
energy consumption minimization, 7-5
minimum-cost forwarding protocol, 7-13–7-14
minimum cost path approach, 7-5
MTE algorithm, 7-16
SAR protocol, 7-14–7-15
QoS-enabled routing protocols
RPAR protocol, 7-31–7-33
SPEED, 7-6
for WSNs, 7-31
topology control protocols, 7-33
GAF protocol, 7-34–7-35
Span protocol, 7-35
STEM protocol, 7-35–7-36
Entity description, 4-12–4-14
Environmental applications, wireless sensor networks, 3-4
EnviroTrack programming system, 12-14
EnviSense, screenshot, 12-33
EPA protocol, *see* Ethernet for Plant Automation (EPA) protocol
Error control strategies, data-link layer, 4-7, 4-12
Error detecting and correcting codes, 24-12; *see also* Forward error correction (FEC) technique
ESP, *see* Electronic stability program
EtherCAT
performance indicators, 21-18
timing, 21-16
Ethernet, 20-4
application layer protocols, OPC, 23-18–23-19
fieldbus combinations, structures of, 20-52
hub-based office network (CSMA/CD)
binary exponential backoff (BEB) algorithm, 23-15–23-16
control application, 23-16
frame format, 23-15
operation, 23-15
industrial network, 23-17–23-18
Internet-based solutions, 20-3
MAC level, 20-55
switched network (CSMA/CA), 23-16–23-17
switching technology, 20-53
TCP/IP, 20-7
two-node network delay, 23-25–23-26
Ethernet for Plant Automation (EPA) protocol
application process, 21-14
performance indicators, 21-14
timing, 21-12
EtherNet/IP
definition, 21-7
performance indicators, 21-8
Ethernet Powerlink (EPL)
definition, 21-10
timing and performance indicators, 21-11
European installation bus (EIB), 20-24
Event-triggered communication, 14-1
EYES project architecture, system abstraction layers
distributed services layer, 4-10
sensor and networking layer, 4-9–4-10
EYES sensor node, 4-2

F

Factory instrumentation protocol (FIP)/WorldFIP, 15-13, 24-13–24-14
FAF, *see* Frame acceptance filtering function
FANs, *see* Field area networks
Fault containment region (FCR), 14-3
FDT/DTM, *see* Field device tool/device type manager
FGL, *see* Fine-grained localization
FHSS, *see* Frequency hopping spread spectrum
Field area networks (FANs), 1-2, 20-5
Field bus data exchange format (FIBEX), 16-15
Fieldbus system
application development
advantages, 20-4, 22-11–22-12
ANSI/ISA-88.01-1995 hierarchical model, 22-12
areas, 20-20
cluster configuration description (CCD) format, 22-14
layer, 20-25
modeling and design approach, 22-11
PROFIBUS DP, mapping of function blocks, 22-13
scheduling approaches, 22-13
automation concept, integrative part, 20-3
automation pyramid, 20-4
building and home automation, 20-58
communication paradigms
client–server model, 20-36–20-37
properties, 20-36
publisher–subscriber model, 20-37–20-39
configuration interfaces
application configuration and upload, 22-15–22-16
hardware configuration, 22-14–22-15
plug and participate, 22-15
ControlNet and P-NET, 20-24
data link layers, 20-24
developers, 20-25
evolution milestones, 20-7, 20-9
fieldbus management, 20-42–20-43
historical roots, 20-6
industrial and process automation, 20-57
industrial Ethernet
real-time capabilities, 20-51
UDP/TCP/IP, 20-52
IP-based networks, 20-46
layer structure, 20-23
maintenance

replacement node, 22-18–22-19
schemes and steps, 22-18
management interfaces
calibration, 22-17–22-18
monitoring and diagnosis, 22-17
medium access control, 20-27–20-35
PLC-based automation systems, 20-28
polling, 20-28–20-30
random access, 20-34–20-35
strategies, 20-27
time-slot-based methods, 20-32–20-34
token passing, 20-30–20-32
networking networks
gateways, 20-49–20-51
Internet interconnection, 20-46
protocol tunneling, 20-46–20-49
network interconnections, 20-45
network topologies, 20-25–20-27
Ethernet nodes, 20-25
Ethernet switch, 20-25
SERCOS, 20-26
NOAH approach, 20-43–20-45
OSI layers, interoperability and profiles, 20-39–20-42
OSI model, 20-23–20-27
presentation layer, 20-25
PROFINET, 20-26
protocols, 20-42
DeviceNet, 26-12–26-13
Profibus DP, 26-11–26-12
IEEE 802.15.4, 26-16–26-17
standardization project, 20-10
time division multiple access strategy, 20-28
topological network structures, 20-26
traffic characteristics
classes and typical properties, 20-21
management data, 20-21–20-22
parallel traffic, 20-22
process data, 20-21
requirements, 20-20
Field device tool/device type manager (FDT/DTM), 22-9
Field multiplexers, 20-4
Fine-grained localization (FGL), 6-21–6-22
FLAMA protocol, *see* Flow-aware medium access (FLAMA) protocol
Flexible time division multiple access (FTDMA), 13-9
FlexRay (communication protocol), 1-4, 16-4, 19-3
application signals, 16-11
automotive requirements
automotive industry, 16-1
cost factor, 16-2–16-3
future proof, 16-2–16-3
migration, 16-2
bus guardian, 16-10–16-11
communication controllers (CC), supports, 16-3
clock synchronization, 1-6, 16-5, 16-7
communication cycle, 16-4
configurations, 16-7
data transmission, frame format, 16-9
development, 16-3
dynamic segment, 1-5–1-6, 13-10, 16-6
ECU, 16-3
electronic technologies, 16-1
faulty controllers, 16-8
information domains, 16-12, 16-13
MAC protocol, 13-10
media access strategy, 16-4
network, 13-9, 16-3
idle time, 16-5, 16-6
topologies support, 1-6, 16-10
OEM-supplier development process, 16-13
physical layer supports, 16-10
series application, 16-11
slot ID, 16-11
start-up process, 16-9
start-up service, 16-8
static segment, 1-5
TDMA communication technologies, 16-4, 16-11–16-12
FlexRay Consortium, 16-1
Flow-aware medium access (FLAMA) protocol, 8-11
Forward error correction (FEC) technique, 24-8, 24-12
Frame acceptance filtering function (FAF), 14-13
Frequency hopping spread spectrum (FHSS), 24-8–24-9
Frequency noise, 5-3
FTDMA, *see* Flexible time division multiple access
FullCAN implementations, 15-13–15-14
Functional specification exchange (FSX) format, 18-12

G

GAF protocol, *see* Geographic-adaptive fidelity protocol
GANs, *see* Global area networks
GEAR protocol, *see* Geographic and energy-aware routing protocol
Generalized projections
distributed algorithms
ring algorithm, 9-12–9-13
star algorithm, 9-16–9-18
spectrum analysis
convex feasibility problem, 9-10
generalized distance, 9-10–9-11
minimization problem, 9-11–9-12
General purpose interface bus (GPIB), 20-7
Geographic-adaptive fidelity (GAF) protocol
node ranking, 7-35
state transitions of, 7-34
Geographical clustering, 4-10–4-11
Geographic and energy-aware routing (GEAR) protocol, 7-30–7-31
Global area networks (GANs), 20-4
Global positioning system (GPS) navigation, 13-14
Gossiping protocol, 7-2
GPIB, *see* General purpose interface bus
GPS navigation, *see* Global positioning system navigation
Greedy perimeter stateless routing (GPSR), 12-31

H

Handshaking technique, 23-9–23-10
Hard real-time systems, 1-2
Hardware abstraction layer (HAL)
  microcontrollers, software development, 12-2
  operating system, 12-3
  optional environment, 12-29
Hardware architecture
  clock time, 5-4
  ECU level, 18-3
  system and network level, 18-2
Heating, ventilation, and air-conditioning (HVAC) systems, 29-1
HEED protocol, *see* Hybrid energy-efficient distributed clustering protocol
High-end sensor nodes, 4-4
HIS flash driver, 18-9
Hop count
  DV-hop method, 6-7
  between several nodes, 6-6
Human machine interface (HMI), 13-2
Hybrid energy-efficient distributed clustering protocol, 7-20–7-21
Hybrid wired/wireless real-time networks
  device implementations
    data-link layer, 26-7–26-8
    higher protocol layers, 26-8–26-9
    physical layer, 26-6–26-7
    wireless extensions, 26-9–26-11
  DeviceNet and EtherNet/IP, 26-4–26-5
  fieldbuses
    DeviceNet, 26-12–26-13
    Profibus DP, 26-11–26-12
  IEEE 802.11, 26-5–26-6
  IEEE 802.15.4, 26-6, 26-16–26-17
  Profibus DP and Profinet IO, 26-3–26-4
  protocol data units (PDUs), 26-8
  real-time Ethernet (RTE)
    EtherNet/IP, 26-15–26-16
    Profinet IO, 26-14–26-15

I

Identifier extension (IDE), 15-7
IDSQ, *see* Information-driven sensor querying
IDU, *see* Interface data unit
IEC, *see* International Electrotechnical Commission
IEEE 802.11 standards
  authentication processes, 25-13–25-14
  carrier sense multiple access (CSMA), 25-9
  carrier-to-interference ratio (CIR), 25-14
  cellular functions, 25-14–25-15
  DSSS mode, 25-10–25-11
  infrared (IR), 25-9
  medium access control (MAC), 25-9
  OFDM, 25-11–25-13
  PHY Modes, 25-12–25-13
  QoS function, 25-9–25-10
  WLAN PCMCIA card, 25-11
IEEE 802.15.4 standards, 29-26–29-27
IEEE 802.15.4/ZigBee, 25-15–25-17
IFS, *see* Interface file system
IGMP, *see* Internet group management protocol
Ill-posed linear operator equations, regularization solution
  quasi-solution method, 9-10
  residual method, 9-9
  Tikhonov's method, 9-8–9-9
In-car embedded networks
  low-cost automotive networks
    fieldbus networks, 13-11
    LIN network, 13-12–13-13
    TTP/A network, 13-13
  multimedia networks
    IDB-1394 network, 13-14
    MOST network, 13-13–13-14
  priority buses
    CAN network, 13-5–13-8
    J1850 network, 13-8
    vehicle area network, 13-8
  time-triggered networks
    communication networks, 13-8
    FlexRay protocol, 13-9–13-10
    TTCAN protocol, 13-10–13-11
Industrial environments
  ad hoc networks, 25-4–25-5
Bluetooth technology (BT)
  asynchronous connection-less (ACL), 25-6
  factory floor units, 25-8
  forward error correction (FEC), 25-5
  frequency hopping (FH), 25-4–25-5
  machine-to-machine communication, 25-7
  master–slave concept, 25-5
  physical layer, 25-5
  piconet and slaves, 25-5–25-6
  sophisticated approach, 25-8
  synchronous connection oriented (SCO), 25-6–25-7
  triple transmission rates, 25-7
cellular systems, 25-3–25-5
IEEE 802.11 standards
  authentication processes, 25-13–25-14
  carrier sense multiple access (CSMA), 25-9
  carrier-to-interference ratio (CIR), 25-14
  cellular functions, 25-14–25-15
  DSSS mode, 25-10–25-11
  infrared (IR), 25-9
  medium access control (MAC), 25-9
  OFDM, 25-11–25-13
  PHY Modes, 25-12–25-13
  QoS function, 25-9–25-10
  WLAN PCMCIA card, 25-11
IEEE 802.15.4/ZigBee
  beacon interval and super frame, 25-16–25-17
  CSMA/CA, 25-16
  full function device (FFD), 25-15
  guaranteed time slots (GTS), 25-16
  reduced function device (RFD), 25-15
mobile ad hoc network (MANET), 25-4
network applications
  networks for control, 23-28–23-29
  networks for diagnostics, 23-29

networks for safety, 23-29–23-30
reconfigurable factory testbed (RFT), 23-30–23-33
network differentiation
Ethernet application layer protocol impact, 23-18–23-19
Ethernet-based networks, 23-14–23-18
network categorization, 23-12
random access with collision arbitration, 23-13–23-14
time-division multiplexing, 23-13
wireless networks, 23-19–23-24
network parameterization
delay and jitter, 23-6–23-9
network QoS vs system performance, 23-11–23-12
speed and bandwidth, 23-4–23-6
wired and wireless QoS metrics, 23-9–23-11
WPAN–WLAN coexistence, 25-17–25–18
frequency domain, 25-18
space, 25-19
time domain, 25-19
WSNs, WPAN and WLANs, 25-3
INET, *see* Internetworking framework
Information clustering, 4-11
Information-driven sensor querying (IDSQ), 7-12
In-network processing, 4-24, 4-26–4-28
Input_jitter, 19-9
Intelligent devices, 5-1
Intelligent warehouse, 3-4
Interaction layer protocol data unit (I-PDU), 13-18, 13-19
Interface data unit (IDU), 20-14
Interface file system (IFS), 14-8, 22-5–22-7
International Electrotechnical Commission (IEC)
definition, 20-2
IEC 61158 standard, 21-3
performance indicators, 21-5–21-6
profiles and CPF lists, 21-3
structure, 21-2–21-3
user-application requirements, 21-3–21-5
International Organization for Standardization (ISO), 18-2
CAN networks, 18-9
client/server diagnostic services, 18-11–18-12
transport layer
frame format, 18-11
transmission sequence, 18-10
International Society of Automation (ISA100), 27-18
architecture, 27-22–27-23
functions, 27-23–27-24
protocol stack, 27-24
security, 27-24
Internet group management protocol (IGMP), 26-5, 26-16
Internetworking framework (INET), 12-30
Interoperability definitions, 20-40
Interval-based communication scheduling protocol, 4-29
Inverse problems, 9-6
I-PDU, *see* Interaction layer protocol data unit
IPv6 over low-power wireless personal area networks (6loWPAN), 27-18
ISO, *see* International Organization for Standardization
Iterative methods, 6-23–6-24

J

Jamming, wireless communication channel, 10-5
JasPar, 18-26–18-27
Jitter, 19-9–19-10
J-Sim, 12-31

K

Key management
alternatives
LEAP, 10-16–10-18
probabilistic key management scheme, 10-18–10-23
for authenticating broadcast-like communications, 10-9
for sensor network requirements, 10-15–10-16
tasks, 10-14–10-15
KNX standards
data exchange nodes, 29-24
design, 29-22
EIBnet/IP routing, 29-23
group objects, 29-24
node architecture, 29-24–29-25
specifications, 29-23
twisted-pair (TP) cabling, 29-23
Kokyu dispatcher in nORB, 2-6, 2-8, 2-10
Konnex, 20-24

L

LAN, *see* Local area network
Layered protocol stack, architecture representation
disadvantages, 4-12
management planes, 4-6
network layers, OSI model, 4-5
application layer, 4-9
data-link layer, 4-7–4-8
network layer, 4-8
physical layer, 4-6–4-7
transport layer, 4-8
LDF, *see* LIN description file
LEACH-C protocol, 7-20
LEACH protocol, *see* Low energy-adaptive cluster hierarchy protocol
LEAP, *see* Localized encryption and authentication protocol
Lian curve, 23-24
Lighthouse localization system, 6-7–6-8
Lightweight medium access, 8-12
LIN description file (LDF)
centric process flow, 17-8
debugging, 17-7
ECU, software, 17-5
file generation, 17-11
system definition tool, 17-6
LLC, *see* Logical link control
LMAC, *see* Lightweight medium access
Local area network (LAN), 1-2, 20-2
Local interconnect network (LIN), 13-3
basics, 17-3
bus properties, 17-2
bus-protocol, 17-1

configuration file, 17-8
diagnostics, and emulation tool, 17-13
emulation control files, 17-12
frame format, 13-12–13-13, 17-5
GO feature, 17-12
GO—graphical objects, 17-13
interframe space, 17-4
logical level definitions, 17-4
mentor graphics LIN tool-chain, 17-7
network architect, 17-7–17-8
data administered, types, 17-8
global objects, definition of, 17-9
synthesis tool, 17-7
network design process, elements, 17-5
open communication standard, 17-1
physical layer, 17-3–17-4
protocol, 17-4–17-5
single wire network, 17-3
Spector—test tool, 17-12–17-13
subnetwork, 17-11
target image building process, 17-12
target package
automatic code generation capability, 17-9
embedded software, Volcano tool chain, 17-9
frame packing, 17-10
network definition, 17-10
timing model, 17-11
vehicles, 17-2
workflow, 17-5, 17-6
Localization process
distance estimation
hop count, 6-6–6-7
lighthouse localization system, 6-7–6-8
minimal transmission power, 6-12
neighborhood intersection, 6-9–6-10
received signal strength, 6-10–6-11
round trip time (RTT), 6-5–6-6
symmetric double-sided two-way ranging, 6-6
three/two neighbor algorithm, 6-8–6-9
time difference of arrival, 6-4–6-5
time of arrival, 6-2–6-4
for network protocols, 6-2
Localization systems, resource requirements, 6-2
Localized encryption and authentication protocol (LEAP)
key establishment procedure, neighboring nodes, 10-16–10-17
security aspects, 10-17–10-18
Location-based routing protocols
GEAR protocol, 7-30–7-31
location information, 7-6
MECN protocol, 7-28–7-30
Logical clock, processing, 2-11–2-12
Logical link control, 15-4
Logical time, 2-11
LonBuilder, 1-13
LonTalk protocol, 1-12
LonWorks, *see* EIA-709
LonWorks standards
control network protocol (CNP), 29-20–29-21
IP tunneling method, 29-20
multicast services, 29-21
node, architecture, 29-21–29-22
reliable transmission mode, 29-21
Low-end sensor nodes, 4-4
Low energy-adaptive cluster hierarchy (LEACH) protocol
cluster-head election mechanism, 7-18–7-19
design principles, 7-17
energy efficiency, 7-19
extension, 7-19–7-20
first-order radio model, 7-17–7-18

M

MAC protocol, *see* Medium access control protocol
Management information base (MIB), 20-44
Manufacturing automation protocol (MAP), 20-8
Manufacturing message specification (MMS), 20-8
Master–slave (MS) network, 23-7–23-8
MATÉ architecture, 12-9
Maximum energy cluster-head (MECH) protocol
forwarding phase, 7-22
vs. LEACH approach, 7-21–7-22
Maximum lifetime routing protocol
energy consumption between nodes, 7-16
vs. MTE algorithm, 7-16
MCUs, *see* Microcontroller units
MECH, *see* Maximum energy cluster-head (MECH) protocol
MECN protocol, *see* Minimum energy communication network protocol
Media independent interface (MII), 14-8
Media oriented system transport (MOST), 13-3
Medium access control (MAC) and link layer protocols
channel errors and channel variation, 24-11–24-13
problems
exposed terminal scenario, 24-10
hidden terminal scenario, 24-9–24-10
solutions
busy tone solutions, 24-10
RTS/CTS protocol, 24-11
Medium access control (MAC) protocol, 12-2, 13-5
classification
B-MAC protocol, 8-8–8-9
CC-MAC protocol, 8-14–8-16
contention-based protocols, 8-3–8-4
crankshaft protocol, 8-13–8-14
DMAC protocol, 8-8
FLAMA protocol, 8-11
LMAC protocol, 8-12
PAMAS protocol, 8-4–8-5
PEDAMACS protocol, 8-10
PMAC protocol, 8-13
schedule-based protocols, 8-4
Sift protocol, 8-11–8-12
S-MAC protocol, 8-5–8-7
T-MAC protocol, 8-7–8-8

TRAMA protocol, 8-10–8-11
WiseMAC protocol, 8-9–8-10
Z-MAC protocol, 8-12–8-13
design issues
code size and memory requirements, 8-1
network lifetime, 8-1
throughput and bandwidth utilization, 8-2
energy waste, wireless networks, 8-2
performance metrics, 8-3
Membership service, TTP/C, 1-5
Memory usage, 17-8
Message flow architecture, nORB, 2-9–2-10
MH-MAC, *see* Mobility-adaptive hybrid MAC
MIB, *see* Management information base
Microcontroller units, 18-3
central unit, 12-2
characteristics of, 12-3
Contiki, 12-18
port pin, 18-23
sensor node, 12-2
software development for, 12-2, 12-3
MicroQoSCORBA, 2-15
Microtick, 16-7
Middleware
design
application domain constraints, 2-3–2-4
implementation challenges, 2-4–2-5
layer
automotive electronic systems, 13-14
AUTOSAR consortium, 13-16
inter-domain communications, 13-15
OSEK/VDX communication layer, 13-16
TITUS/DBKOM communication, 13-15
linking applications and networks, 12-5
multiple DOC, 2-6
special-purpose (*see* Special-purpose middleware)
support for temporal coordination in simulation environment, 2-8
MII, *see* Media independent interface
Middleware linking applications and networks (MiLAN)
bandwidth, 12-12
cost values, 12-13
middleware concept, 12-12
strategies, 12-13
Military applications, wireless sensor networks, 3-4
MIL STD 1553 bus, 20-8
Minimal transmission power (MTP), distance estimation, 6-12
MiniMAP, 20-8
Minimum-cost forwarding protocol
fault tolerance, 7-14
message delivery through minimum cost path, 7-13–7-14
Minimum energy communication network (MECN) protocol
received power, RF systems, 7-29
synchronous communications, 7-28
MMAC protocol, *see* Mobility-adaptive collision-free MAC protocol
MMSN protocol, *see* Multifrequency MAC for wireless sensor networks protocol
MMS, *see* Manufacturing message specification
Mobility-adaptive collision-free MAC (MMAC) protocol
cluster-based approach, 8-18
location information, 8-17
scheduled-access and random-access time-slots, 8-17
Mobility-adaptive hybrid MAC (MH-MAC), 8-18
Mobility-aware MAC for sensor networks protocol (MS-MAC), 8-16–8-17
Mobility management plane, 4-5
Modbus protocol, 29-17
Modbus/TCP, 21-7
Modern computer networks, foundations, 20-7
Modulation schemes, 4-6
MOST, *see* Media oriented system transport
Mote (wireless sensor node), 9-1–9-2
MS-MAC protocol, *see* Mobility-aware MAC for sensor networks protocol
Multichannel LMAC, 8-20
Multichannel MAC, 8-20
Multichannel protocols
channel assignment, 8-19
hybrid approaches, 8-18
LMAC protocol, 8-20
MAC protocols and, 8-19
MCMAC protocol, 8-20
MMSN protocol, 8-19–8-20
Multifrequency MAC for wireless sensor network (MMSN) protocol
assignment strategies, 8-19
unicast packets detection, 8-20
Multi-hop communication, 10-3
mySQL, 12-33

N

Neighborhood intersection distance estimation scheme (NIDES), 6-9–6-10
Networked control system (NCS) characterisation
analytical perspective
control and safety functionality, 23-27
cost factor, 23-26
weighted analysis, 23-26
experimental perspective, 23-25–23-26
theoretical perspective, 23-24–23-25
Networked embedded automotive applications, *see* Automotive networked embedded systems
Networked embedded system
building automation (*see* Building automation systems)
characterization, 21–2-2
definition, 1-1
design dimensions, 2-2–2-3
design methods, 1-3
hardware infrastructure, 2-2
industrial automation

challenges, 1-9
Ethernet, 1-7–1-8
fieldbus technology, 1-7
operational security requirements, 1-8–1-9
large-scale, 2-4
memory-constrained, 2-5
middleware (*see* Middleware)
ORB middleware
message flow architecture, 2-9–2-10
message formats, 2-8–2-9
object adapter, 2-9
real-time priority propagation, 2-11
simulation support, 2-11–2-13
time-triggered dispatching mechanisms, 2-10
reasons for emergence, 1-1
safety-critical applications, 1-3
Networked sensing system, 9-1
Network idle time, 16-6
Networking systems
architectural layer model, 12-3–12-4
sensor node, 12-2
layers
functions, 4-8
management planes, 4-5
Network management (NMT) state machine, 15-32
Network-oriented application harmonization (NOAH)
ACORN and RACKS, 20-43
management information base, 20-44
system architecture, 20-44
Network time protocol (NTP), 5-6
Network time unit (NTU), 14-23
Neuron C, 1-13
Neuron chips-based controllers, 1-13
NmLimpHome, 18-6
N-modular redundancy (NMR), 14-5
NOAH, *see* Network-oriented application harmonization
NodeBuilder, 1-13
Non-return-to-zero (NRZ) bit representation, 13-5
nORB middleware
bottom-up compositional approach, 2-7
communication between, 2-7
design recommendations and trade-offs
data types, 2-13–2-14
engineering life cycle issues, 2-15
messaging protocol, 2-14
safety and liveness requirements, 2-14
message flow architecture, 2-9–2-10
message formats, 2-8–2-9
object adapter, 2-9
real-time priority propagation, 2-11
simulation support, 2-11–2-13
time-triggered dispatching mechanisms, 2-10
NTP, *see* Network time protocol
NTU, *see* Network time unit

O

oBIX, *see* Open building information exchange
Object management group (OMG), 14-11
ODVA, *see* Open DeviceNet Vendor Association
OEM-supplier development process, 16-14
OGC, *see* Open Geospatial Consortium Inc.
OLE for process control (OPC), 20-42
OMG, *see* Object management group
Open building information exchange (oBIX), 29-29–29-30
Open DeviceNet Vendor Association (ODVA), 15-3
Open Geospatial Consortium (OGC) Inc., 12-20
Open systems interconnection (OSI) model
answered service, 20-16
communication services
important units, 20-13–20-14
service access point, 20-13–20-16
concepts, 20-10
confirmed service, 20-15
definition, 20-10
fieldbus systems, 20-23–20-25
layer structure
application layer, 20-13
data link layer, 20-12
network layer, 20-12
physical layer, 20-11
presentation layer, 20-13
session layer, 20-13
transport layer, 20-12
locally confirmed service, 20-15
protocol layers, 4-5, 15-3
application layer, 4-9
data-link layer, 4-7–4-8
network layer, 4-8
physical layer, 4-6–4-7
transport layer, 4-8
Optimal signals packing, 17-6
Optimization-based routing protocols, 7-4
EAR protocol, 7-15–7-16
energy consumption minimization, 7-5
minimum-cost forwarding protocol, 7-13–7-14
minimum cost path approach, 7-5
minimum transmitted energy (MTE) algorithm, 7-16
SAR protocol, 7-14–7-15
ORB-style middleware, development, 2-5
Orthogonal frequency division multiplexing (OFDM), 24-9, 25-11–25-13
OSEK NM message format, 18-6
OSEK NM state automaton, 18-5
OSEK OS task states, 18-4
OSEKtime FTCom, 18-7
OSEKtime task states, 18-7
OSEK/VDX communication layer, 13-16, 16-15
OSEK/VDX standard
HIS working group
CAN driver, 18-9
flash driver, 18-9
I/O library, 18-8–18-9
OSEK OS, 18-8
standardized components, 18-7–18-8
OSEK COM, 18-6
OSEK NM, 18-5–18-6
OSEK OS, 18-4
OSEKtime FTCom, 18-7

OSEKtime OS, 18-6–18-7
standardized components, 18-3–18-4

P

Packet identification field (PID), 18-14
PAMAS, *see* Power-aware multi-access protocol with signaling
PAN, *see* Personal area network
Pattern MAC (PMAC) protocol, 8-13
Pattern matching techniques, 6-24
PCB-level busses, instrumentation, 20-57
PCI, *see* Protocol control information
Process data object (PDO)
communication parameter, 15-29
definition, 15-36
PDU multiplexer module, 18-20
PDU, *see* Protocol data unit
PEDAMACS, *see* Power-efficient and delay-aware MAC for sensor networks
PEGASIS protocol, *see* Power-efficient gathering in sensor information systems protocol
PermaSense project
objective, 11-21
PermaSense system architecture
components, 11-22
current consumption, 11-24
long-term trending of power consumption, DSN testbed, 11-22
memory profiling, 11-23
Personal area network (PAN), 29-26
Phase noise, 5-3
PID, *see* Packet identification field
Ping node scheduling
for active damage detection, 2-3–2-4
performance, 2-4
Plastic optical fiber (POF), 13-14
PLCs, *see* Programmable logic controllers
PMAC protocol, *see* Pattern MAC protocol
P-NET, 21-8
PNO (Profibus Nutzerorganization) WSAN, 27-18
POF, *see* Plastic optical fiber
Point coordination function (PCF), 24-15–24-16
Power-aware multi-access protocol with signaling (PAMAS), 8-4–8-5
Power consumption control, data-link layer, 4-8
Power-efficient and delay-aware MAC for sensor networks (PEDAMACS), 8-10
Power-efficient gathering in sensor information systems (PEGASIS) protocol
chain-based approach, 7-24–7-25
chain-based forwarding, 7-25
data gathering schemes, 7-26
Power management plane, 4-5
Power supply, WSN device
energy conversion
mechanical converters, 27-42
photovoltaic cells, 27-41
thermoelectric converters (TECs), 27-42
energy storage
electrical buffer storages, 27-40–27-41
permanent storages, 27-40
energy transmission
capacitive transmission, 27-42–27-43
inductive transmission, 27-42
optical transmission, 27-43
power management, 27-39–27-40
requirements
energy buffer, 27-38
form factors, 27-39
lifetime, 27-39
power demands, 27-38
Predefined connection set, COBs foreseen, 15-34
Privacy homomorphism-based data aggregation scheme, 10-26–10-27
Probabilistic key management scheme
key pre-distribution phase, 10-18–10-19
path key establishment phase, 10-19–10-22
shared key discovery phase, 10-19
Process data exchange, 29-9
Producer/consumer model, 14-12
PROFINET and EtherNet/IP, 23-17–23-18
PROFINET CBA, 21-14–21-15
PROFINET IO
performance indicators, 21-19
timing, 21-17
Programmable logic controllers (PLCs), 20-4
Protocol control information (PCI), 18-11, 20-14
Protocol data unit, 10-7, 20-13
Proximity-based techniques
bounding box algorithm, 6-20–6-21
centroid localization, 6-16–6-17
nearest beacon, 6-16
range-free localization, 6-19–6-20
weighted CL, 6-17–6-19
Publisher
subscriber model, basic idea, 20-37
subscriber paradigm, push model, 20-38, 20-39
Pulse width modulation (PWM), 18-9
PWM, *see* Pulse width modulation

Q

QoS-enabled routing protocols
RPAR protocol, 7-31–7-33
SPEED, 7-6
for WSNs, 7-31
Quality-of-Service (QoS), 12-2, 13-2
closed/open-loop control, 20-19
definition, 23-4
delay jitter, 20-17
end-to-end delay, 20-17
file transfer, 20-19
frequency, 20-17
hard real-time, 20-18
loss rate, 20-17–20-18
monitoring/logging, 20-19
packet ordering, 20-18
parameters, 20-16
soft real-time, 20-18
time-to-live parameter, 20-17
usage, 20-16
vs. system performance, 23-11–23-12

wired and wireless metrics, 23-9–23-10
Quasi-solution method, 9-10
Query processor, WSN, 4-28

R

Range-free localization algorithm, 6-19–6-20
Rate-based diffusion algorithm, 5-10–5-11
RBS, *see* Reference-broadcast synchronization
Real-time Ethernet (RTE)
IEC standards
IEC 61158, 21-3
performance indicators, 21-5–21-6
profiles and CPF lists, 21-3
structure, 21-2–21-3
user-application requirements, 21-3–21-5
IEEE 802.11
EtherNet/IP, 26-15–26-16
Profinet IO, 26-14–26-15
IEEE 802.15.4, 26-17
modified Ethernet
EtherCAT, 21-17–21-18
PROFINET IO, 21-18–21-19
SEriell Real time COmmunication System Interface (SERCOS), 21-15–21-16
performance indicators, 21-5–21-6
protocol architecture, 20-54
on top of Ethernet
Ethernet for Plant Automation (EPA), 21-13–21-14
Ethernet Powerlink (EPL), 21-10–21-12
PROFINET CBA, 21-14–21-15
time-critical control network (TCnet), 21-12–21-13
on top of TCP/IP protocols
EtherNet/IP, 21-7–21-8
Modbus/TCP, 21-7
P-NET, 21-8
Vnet/IP, 21-9
topologies and structures, 21-5–21-6
Real-time hardware abstraction layer (RTHAL), 14-8
Real-time power-aware routing (RPAR) protocol, 7-31–7-33
Received signal strength (RSS), distance estimation, 6-10–6-11
Receive error count (REC), 15-11
Reconfigurable factory testbed (RFT), multilevel factory networking
control networks, 23-30–23-31
experimental setup, 23-32–23-33
hardware components, 23-30
safety network implementation, 23-30, 23-32
schematic diagram, 23-30–23-31
serial–parallel line component, 23-30
software components, 23-30
trouble-shooting tool, 23-32
Reference-broadcast synchronization (RBS), 5-8
Remote communication, 2-5
Remote procedure calls (RPCs), 12-5
Remote transmission request, 15-7
REQ messages, 7-7–7-8
Request-to-send/clear-to-send (RTS/CTS) protocol, 24-11
Residual method, 9-9
Ring algorithm, 9-12–9-13
Round trip time (RTT), 5-5, 6-5–6-6
RPAR protocol, *see* Real-time power-aware routing protocol
RPCs, *see* Remote procedure calls
RTE, *see* Real-time Ethernet; Run time environment
RTHAL, *see* Real-time hardware abstraction layer
RT publisher–subscriber (RTPS) protocol, 21-7
RTR, *see* Remote transmission request
RTT, *see* Round trip time
Rumor routing protocol
energy efficiency, 7-11
query flooding and event flooding, comparison, 7-10–7-11
Run time environment (RTE), 13-16, 18-16

S

SAE, *see* Society for Automotive Engineers
SAP, *see* Service access point
SAR protocol, *see* Sequential assigned routing protocol
SCADA, *see* Supervisory control and data acquisition (SCADA)
Schedulability analysis, 1-3
SCI, *see* Serial communication interface
SCNM, *see* Slot communication network management
SDS, *see* Smart distributed system
SDS-TWR, *see* Symmetric double-sided two-way ranging
SDU, *see* Service data unit
Secure data aggregation scheme
based on privacy homomorphism, 10-26–10-27
CDA scheme, 10-26
confidentiality, sensor readings, 10-26
data fusion nodes, 10-23
drawbacks, 10-27
security, 10-25
voting scheme, DoS vulnerability, 10-24
witness-based approach, 10-23–10-24
Security clustering, 4-11
Self-organizing medium access control for sensor networks (SMACS), 7-15
Semantic routing tree (SRT), 4-31
SeNeTs
application server, 12-28
communication channels, 12-27
development process, sensor node software, 12-17, 12-18
environment management, 12-29
interface optimization, 12-16
compile time, 12-17
types, 12-17
network server, 12-27–12-28
node application, structure, 12-15
SeA, software layer model, 12-28
secondary transmission channel, 12-27
sensor network structure, 12-15
SNA, software layer model, 12-28

software-layer model, interfaces, 12-16
system architecture, 12-26
wireless sensor nodes, applications, 12-28
Sensor MAC (S-MAC) protocol, 8-5
drawback, 8-7
low-duty-cycle operation, 8-7
major components, 8-6
message passing scheme, 8-6
Sensor model language (SensorML), 12-21
Sensor network encryption protocol (SNEP)
basic trust model, 10-9
CBC-MAC construction for MACs, 10-10–10-11
message authentication codes, 10-10
message formats, 10-11
RC5 algorithm
encryption function, 10-10
parameter configuration of, 10-9
security, 10-10
shared secret, 10-12
Sensor network fusion problem
advantages to formulating, 9-2
topologies for distributed, 9-19
Sensor networks; *see also* Wireless sensor networks
application-specific characteristics, 10-2, 10-3
building blocks needed to setup
application layer, 4-9
data-link layer, 4-7–4-8
network layer, 4-8
physical layer, 4-6–4-7
transport layer, 4-8
clustering, 4-10–4-11
communication, 4-10
data aggregation, 10-23
development, 12-31–12-33
distance estimation of (*see* Distance estimation methods)
DoS concerns
breaking into sensor nodes, 10-4
flooding, 10-6
forwarding and routing, 10-5–10-6
jamming, 10-5
dynamic behavior, 4-11–4-12
EmStar, 12-24–12-26
fusion problem (*see* Sensor network fusion problem)
generalized component interface, 4-16–4-17
heterogeneous, 4-3
higher-level middleware functionality, 12-4
J-Sim, 12-30–12-31
leakage detection, 12-2
motes, 9-1–9-2
network topology, 4-11
programming aspect *vs.* behavioral aspect, 12-4–12-5
propagation delay variation minimization, 5-5–5-6
security, 10-1
objectives, 10-3
protocols for realizing efficient, 10-8–10-9
SNEP, 10-9–10-12
μTESLA, 10-13–10-14
sensor modules, 4-11
sensor nodes, 12-12
simulation tools, 12-5–12-8
software architectures
Contiki, 12-18–12-20
EnviroTrack, 12-13–12-14
MATÉ, 12-8–12-9
MiLAN, 12-12–12-13
SeNeTs, 12-14–12-18
SensorWare, 12-10–12-12
sensor web enablement, 12-20–12-22
TinyDB, 12-10
TinyOS, 12-5–12-8
spectrum analysis
convex feasibility problem, 9-10
generalized distance, 9-10–9-11
minimization problem, 9-11–9-12
power spectrum estimation, 9-5–6
problem formulation, 9-4
sound source, 9-3–9-4
testbeds, 11-9–11-10
tiered architecture, 4-10
time synchronization protocol (*see* Time synchronization protocols)
TinyOS SIMulator, 12-23
topology
base stations, 10-2–10-3
multi-hop communication, 10-3
validation environment, SeNeTs—test, 12-26–12-30
Sensor nodes, 12-1
applications, 12-3
architecture, 12-12
based on OSI layers, 4-5–4-6
communication interface, 4-3
energy efficiency, 4-4
layered protocol stack description of, 4-5–4-9
main components of, 4-4
power source, 4-3
processing unit, 4-2–4-3
protocol place in, 4-12
sensor and actuating platform, 4-2
characterization, 3-3
in clusters, 4-10–4-11
collaboration between, 4-1
distributed fusion of sensor data across, 9-2
dynamic, 4-10
with entity, algorithm running inside, 4-12, 4-14
flexible architecture design, 4-11–4-12
data exchange, 4-16
data types, 4-14
entity description, 4-13–4-14
functions, 4-10
hardware emulation, 12-23
high-end, 4-4
low-end, 4-4
microcontroller central unit, 12-2
modes of operation
initialization mode, 4-4
operation mode, 4-4–4-5
position estimation
importance, 6-1
mathematical methods, 6-2
Quality-of-Service, 12-2
sensor network, 12-22
simulation tools, 12-22
software, 12-3
SRT, 4-31
static, 4-10
structure, 12-2
time difference, 5-1

Sensor planning service (SPS), 12-22
Sensor protocols for information via negotiation (SPIN)
  basic functioning, 7-7–7-8
  messages introduced, 7-7
  metadata negotiation, 7-6
Sensor Web enablement (SWE)
  networks based communication structure, 12-21
  observations and measurements, 12-21
  sensor model language, 12-21
  sensor observation service, 12-22
  sensor planning service, 12-22
  transducer markup language, 12-22
Sequential assigned routing (SAR) protocol, 7-14–7-15
Serial communication interface (SCI), 17-2
SErial Real time COmmunication System Interface (SERCOS)
  communication channels, 21-15
  performance indicators, 21-16
Service access point (SAP), 20-13
Service data unit (SDU), 20-13
Sift protocol, 8-11–8-12
Signal, max_age, 19-9
Simple network management protocol (SNMP), 20-44
Slot communication network management (SCNM), 21-10
S-MAC protocol, *see* Sensor MAC protocol
SMACS, *see* Self-organizing medium access control for sensor networks
Smart distributed system (SDS), 20-8
SNEP, *see* Sensor network encryption protocol (SNEP)
SNMP, *see* Simple network management protocol
Society for Automotive Engineers (SAE), 13-3
Soft real-time systems, 1-2
Software access time, 5-4
Software industry, 17-2
Software mastering tools, multitier model, 12-32
Software programming techniques, 12-4
Software requirement specification (SWRS), 19-12
Span protocol, 7-35
Sparse topology and energy management (STEM) protocol
  assumption, 7-35
  in reactive WSNs, 7-36
  solutions proposed, 7-35–7-36
Special-purpose middleware
  approaches to develop
    bottom-up approach, 2-6, 2-7
    top-down approach, 2-5–2-6
  design
    application domain constraints, 2-3–2-4
    and implementation challenges, 2-4–2-5
Spectrum analysis
  generalized projections
    convex feasibility problem, 9-10
    generalized distance, 9-10–9-11
    minimization problem, 9-11–9-12
  sensor networks
    power spectrum estimation, 9-5–6
    problem formulation, 9-4
    sound source, 9-3–9-4
SPIN, *see* Sensor protocols for information via negotiation
SPS, *see* Sensor planning service
SRR, *see* Substitute remote request
SRT, *see* Semantic routing tree
Standard network variable types, 20-41
Standard software components, 16-15–16-16
Star algorithm, 9-16–9-18
STEM protocol, *see* Sparse topology and energy management protocol
Substitute remote request (SRR), 15-7
Supervisory control and data acquisition (SCADA), 23-2, 23-18, 23-28
Sweeps and ad hoc positioning system, 6-23
SWE, *see* Sensor Web enablement
SWRS, *see* Software requirement specification
Symmetric double-sided two-way ranging (SDS-TWR), 6-6
Systems on a chip, advantages, 1-1

T

TAO, 2-6, 2-8
Task management plane, 4-5
TCnet, *see* Time-critical control network
TCP/IP protocol, 20-54
TCP/UDP/IP
  communication channel, 20-55
  mechanisms, 20-54
TDMA-based protocols, safety-critical applications, 1-3
TDMA, *see* Time division multiple access
TDoA, *see* Time difference of arrival
TDP, *see* Time-diffusion synchronization protocol
TEC, *see* Transmission error count
TEDS, *see* Transducer electronic datasheet
TEEN protocol, *see* Threshold-sensitive energy-efficient sensor network protocol
Telematics functions, 13-3
μTESLA protocol, 10-13–10-14
Thermal noise, 24-7
Three/two neighbor algorithm, 6-8–6-9
Threshold-sensitive energy-efficient sensor network (TEEN) protocol, 7-23-
Tikhonov's method, 9-8–9-9
Time-critical control network (TCnet)
  common memory system, 21-13
  high-speed-transmission-period, 21-12
  performance indicators, 21-13
  timing, 21-11
Time delay and jitter
  definition, 23-6
  pre-and postprocessing times, 23-7

source node waiting time
master–slave network, 23-7–23-8
strobe message configuration, 23-9
timing diagram, 23-6
Time difference of arrival (TDoA), 6-4–6-5
Time-diffusion synchronization protocol (TDP), 5-9
Time division multiple access (TDMA), 13-4,-15-18
communication network, 14-2
strategy, 20-28
Time division multiple access (TDMA), 24-13
Time-division multiplexing (TDM), 23-13
Time of arrival, 6-2–6-4
Timeout-MAC (T-MAC) protocol
early-sleeping problem, 8-7
FRTS mechanism, 8-7
and S-MAC, duty cycles comparison between, 8-8
Time synchronization protocols
basics, 5-3
hardware clock time, 5-4
round-trip time, 5-4–5-5
design challenges, 5-2–5-3
factors influencing, 5-3
propagation delay variation minimization, 5-5
purpose, 5-1
for sensor networks
NTP, 5-6
rate-based diffusion algorithm, 5-10–5-11
RBS, 5-8
TDP, 5-9
TPSN, 5-7
time synchronization for high latency (TSHL), 5-7–5-8
synchronization protocols, 5-6
Time-triggered architecture, 14-1
FTUs, 1-5
nodes, 1-4
system activities within, 2-15
and TTP/A and TTP/C protocols, 1-4–1-5
Time-triggered CAN protocol, 15-21
Time-triggered communication, 14-1
automotive domain, 14-2
dependability concepts
computing system, 14-2
error containment, 14-3
fault containment region, 14-3
fundamental services
clock synchronization, 14-3, 14-4
diagnostic services, 14-5–14-6
fault isolation mechanisms, 14-4–14-5
messages carrying, periodic exchange, 14-4
time-triggered Ethernet
clock synchronization, 14-11–14-12
commercial/prototypical components, 14-12–14-13
fault isolation mechanisms, 14-12
state messages, periodic exchange, 14-12
TTCAN
clock synchronization, 14-10
commercial/prototypical components, 14-11
diagnostic services, 14-11
fault isolation mechanisms, 14-11
state messages, periodic exchange of, 14-10
system matrix in, 14-23
time-triggered operation, 14-24
TTP/A 14-8–14-9
TTP/C network, 14-7–14-8
Time-triggered controller area networks (TTCAN), 1-6, 13-4
basic cycle, 13-11
IPModule, 14-11
TTCAN protocol (*see* Time-triggered CAN protocol)
Time-triggered dispatching, nORB, 2-10
Time-triggered Ethernet node, Soekris 4801, 14-13
Time-triggered message-triggered object architecture, 2-15
Timing analysis, 1-3
Timing constraints, capture, 19-10–19-11
Timing-sync protocol for, sensor networks (TPSN), 5-7
Tiny application sensor kit, 12-33
TinyDB, 4-23
aggregation operations, 4-26
multi-query optimization, event-based queries, 4-28
routing, 4-31
slotted scheduling protocol, 4-29
TinyOS
application, 12-8
architecture, 12-7
components, categories of, 12-7
elemental properties, 12-6
SIMulator, 12-23
software architecture, 12-6
TOSSIM system architecture, 12-24
TinyOS SIMulator (TOSSIM), 12-23
T-MAC protocol, *see* Timeout-MAC protocol
TML, *see* Transducer markup language
TMR, *see* Triple-modular redundancy
ToA, *see* Time of arrival
Toggle snooping, 8-19
Top-down approach, 2-5–2-6
Topology control protocols, 7-33
GAF protocol, 7-34–7-35
Span protocol, 7-35
STEM protocol, 7-35–7-36
TOSSIM, *see* TinyOS SIMulator
TPSN, *see* Timing-sync protocol for sensor networks
Traffic-adaptive medium access
NP, SEP, and AEAs, 8-11
vs. S-MAC, 8-11
TRAMA, *see* Traffic-adaptive medium access
Transducer electronic datasheet (TEDS), 22-9–22-10
Transducer markup language (TML), 12-22
Transmission error count (TEC), 15-11
Transmission media, kinds of, 15-5
Transport layer, 4-8
Triangulation, 6-2, 6-15–6-16
Trilateration, 6-2, 6-16
Triple-modular redundancy (TMR), 14-5
TTA, *see* Time-triggered architecture;

Time-triggered architectures
TTCAN, *see* Time-triggered controller area network
TT communication systems, 13-8
TTDD protocol, *see* Two-tier data dissemination protocol
TTP/A node, 14-10
TTP based on TDMA medium access control technology, 1-4
TTP/C
  network
    fault-tolerant clock synchronization, 14-7
    star topology and bus topology, 14-6
  protocol
    and FlexRay protocol, 1-6
    membership service, 1-5
    synchronous TDMA medium access control scheme, 1-5
    in TTA, 1-4
TTP monitoring node, 14-8
Two-tier data dissemination (TTDD) protocol, 7-13
Two-way message handshake, 5-7

U

Ubiquitous computing, 3-2
Ubiquitous CORBA projects, 2-15
UCI-Core approach, 2-6
Unacknowledged segmented data transfer (USDT), 18-9
Universal asynchronous receiver transmitter (UART), 14-8, 15-7, 17-2
Universally Interoperable Core (UIC), CORBA specialization, 2-15
User datagram protocol (UDP), 20-12, 23-21

V

VAN, *see* Vehicle area network
VDX, *see* Vehicle Distributed Executive
Vehicle area network, 13-8
Vehicle distributed executive, 18-1
Verilog, *see* VHSIC hardware description language
VHSIC hardware description language, 12-7
v_imf_queued(), 19-8
v_imf_rx(), 19-8
v_imf_tx(), 19-8
Virtual local area networks (VLANs), 26-5, 26-16
Virtual private network (VPN), 23-10
VNA, *see* Volcano network architect
Vnet/IP, 21-9
Volcano
  back-end tool, 19-12
  boolean/integer signals, advantage of, 19-5
  concepts, 19-4
  configuration, 19-17
  configuration files
    ECU's memory, 19-20
    executable binary code, 19-19
    files, types of, 19-17–19-18
    fixed information, 19-18
    network information, 19-18
    private information, 19-18
    target information, 19-18
    target software, 19-19
  definition, 19-2
  ECUs, network development, 19-18
  flag, 19-6
  frame modes, 19-7
  immediate frames, 19-6–19-7
  network frames, 19-6
  network interface, 19-7
  publish/subscribe model, 19-5
  resource information, 19-8
  signal, declaration of, 19-10
  timeout, 19-6
  timing guarantees, 19-4
  timing model, 19-8–19-9
  update bit, 19-5–19-6
  workflow, 19-18–19-20
Volcano API, CAN/LIN controllers, 19-7
Volcano network architect
  automotive groups tool, 19-12
  car OEM tool chain, 19-12
  consistency checking, 19-14
  database, 19-13
  data configuration, 19-13
  data export, 19-17
  data, gatewaying of, 19-17
  data import, 19-17
  electrical architectures, 19-11
  global objects, 19-13
  multiprotocol support, 19-16–19-17
  structure of, 19-14
  timing analysis/frame compilation, 19-14–19-15
  version and variant handling, 19-13
  volcano filtering algorithm, 19-15–19-16
Volcano processing calls, execution time, 19-8
Volcano thread-of-control, 19-7–19-8
Volvo Car Corporation, 19-2
Volvo S80 main networks, 19-2

W

WCL, *see* Weighted centroid localization
Weighted centroid localization, 6-17–6-19
Wireless communications
  integration of, 13-2
  properties of, 20-55
Wireless Ethernet/IEEE 802.11
  description, 24-15–24-16
  real-time transmission, 24-16
Wireless fieldbus
  mobility support, 24-5
  real-time error-prone channel transmission, 24-3–24-4
  security aspects and coexistence, 24-5
  state of the art
    controller area network (CAN), 24-13
    FIP/WorldFIP, 24-13–24-14
    IEC fieldbus, 24-14–24-15
    PROFIBUS, 24-14
  system aspects, 24-3
  wired-wireless station integration
    classification, 24-4–24-5
    hybrid system requirements, 24-4
WirelessHART
  architecture, 27-20
  mechanisms, 27-21
  protocol stacks, 27-21
  security, 27-22
  WSN standard, 27-17

Wireless interface for sensors and actuators (WISA)
application and performance
automotive stamping, 28-23–28-24
characteristic figures, 28-26–28-27
discrete factory automation, 28-24–28-25
fieldbus, 28-23
latency measurements, 28-23
TDMA real-time system, 28-26
WISA-POWER and WISA-COM technologies, 28-21, 28-23
automation
domains, 28-1
hierarchy, 28-1–28-3
design
antenna diversity and switching, 28-8
frequency hopping, 28-6, 28-8–28-9
medium access and retransmission, 28-6–28-7
multicell operation, 28-8–28-9
implementation
base station, 28-9–28-10
Bluetooth-equipped laptop, 28-15, 28-17
coexistence, 28-10–28-13
industrial electromagnetic interference, 28-14–28-15
laptop PC, 28-15–28-16
performance measurements, 28-13–28-14
sensor module, 28-10
requirements and system specifications
average automotive assembly plant, 28-3
communication requirements, 28-5
latency distribution, 28-4–28-5
protocols, 28-5–28-6
roundtable production, 28-4
wireless requirements, 28-3–28-4
wireless power subsystem
concepts and technologies, 28-17–28-18
figure of merit, 28-18–28-20
magnetic supply, 28-20
omnidirectional receiver structure, 28-21
resonant, medium-frequency power supply, 28-20
rotating field, 28-21–28-22
Wireless local area network (WLAN)
authentication processes, 25-13–25-14
IEEE 802.11b WLAN PCMCIA card, 25-11
IEEE 802.11 standard, 24-15–24-16
industrial communications and fieldbus, 24-3–24-5
MAC concept, 25-9–25-10
mobile stations (MS), 25-9
physical layer
transmission effects, 24-7–24-8
wireless transmission techniques, 24-8–24-9
wave propagation effects, 24-6–24-7
WPAN–WLAN coexistence, 25-17–25-19
Wireless microsensor networks, 12-1
Wireless networks
Bluetooth
advantages, 23-22
determinism characterization, 23-22–23-24
drawbacks, 23-22
Ethernet (CSMA/CA)
frame times and delays, 23-21
maximum throughputs, 23-20
packet delay distribution, 23-21
timing diagram, 23-20
user datagram protocol (UDP), 23-21
Wireless personal area network (WPAN), 25-3
user-data rates, 25-2
WLAN–WPAN coexistence, 25-17–25-19
Wireless sensor/actuator network, 1-2
Wireless sensor networks, 12-4, 25-3; *see also* Sensor network
AI-LMAC protocol, 4-29
application areas, 3-1, 10-2
applications and protocols, 12-22
approaches to relaying data, 7-2
architectures of
EYES project approach, 4-9–4-11
protocol stack approach, 4-5–4-9
causes of energy waste, 8-2–8-3
characterization, 3-3
classical, 27-2–27-3
classification
area of application, 3-4–3-5
complexity of networks involved, 3-4
communication patterns, 8-2
communication standards
6loWPAN, 27-18
comparison, 27-17
ISA100, 27-18
PNO WSAN, 27-18
proprietary solutions, 27-18–27-19
WirelessHART, 27-17
ZigBee, 27-17
components, 27-3
cross-layering, 8-21
data-centric addressing scheme, 4-30
deployment, 12-32
design challenges
bandwidth, 3-8
diversity and dynamics, 3-8
energy consumption, 3-7, 7-3
lifetime of network, 3-7
redundancy, 3-9
resources availability, 3-6
security, 3-9–3-10
signal processing algorithms, 3-8–3-9
timing and synchronization block, 3-9
development aspects, 27-10
development process
characteristics, 27-6–27-7
communication technology, 27-8–27-9
diverse requirements, 27-8
schematic representation, 27-9

suppliers and components, 27-9
DUST Networks SmartMesh MoC solution, 27-16
distributed data extraction system
DoS concerns, 10-5
energy-saving routing protocols (*see* Energy-saving routing protocols)
feature, 12-3
goal, 4-8
industrial automation applications
characterization, 27-5
ISA-100 usage classes, 27-5–27-6
process attributes, impacts, 27-6–27-7
localization algorithms
assumption-based coordinates algorithm, 6-23–6-24
bounding box algorithm, 6-20–6-21
coarse-grained localization, 6-16–6-17
convex position estimation, 6-22–6-23
DLS algorithm, 6-22
fine-grained localization, 6-21–6-22
nearest beacon, 6-16
pattern matching, 6-24
range-free localization, 6-19–6-20
sweeps and ad hoc positioning system, 6-23
triangulation, 6-15–6-16
trilateration, 6-16
weighted centroid localization, 6-17–6-19
low-power design
basic principles, 27-25–27-26
energy efficient protocols, 27-28–27-29
human–machine interface (HMI), 27-27–27-28
sensor node state diagram, 27-26
sleep modes, 27-26–27-27
MAC protocols (*see* MAC protocols for WSNs)
mobility support
MH-MAC protocol and, 8-18
MMAC protocol, 8-17–8-18
MS-MAC protocol, 8-16–8-17
modularity
aspects/goals, 27-34
HW *vs.* SW components, 27-34–27-35
interfaces, 27-36
lifetime, 27-36
single *vs.* multiple chips, 27-35–27-36
subsystem behavior, 27-37
multichannel protocols (*see* Multichannel protocols for WSNs)
packaging
battery design, 27-32
cooling mechanisms, 27-32
environmental considerations, 27-31–27-32
form factors, 27-32–27-33
hazardous environments (EX), 27-30–27-31
IP (International Protection) rating, 27-29–27-30
positioning systems
Active Badge system, 6-13
design considerations of, 6-14–6-15
importance of, 6-12
RADAR and GPS, 6-14
power supply
power management, 27-39–27-40
requirements, 27-38–27-39
sources and technology, 27-40–27-43
proactive, 7-3
protocol stack approach to build, 4-5
query processor, 4-28
reactive, 7-3
(re)-definition, 27-3–27-4
research projects, 3-5
routing protocols, performance metrics
energy efficiency, 7-3
network partitioning, 7-4
routing scheme, 7-2
countermeasures regarding, 10-6
direct/multi-hop, 7-3
using breadth first spanning tree, attacks on, 10-6–10-7
security objectives, 10-3–10-4
SeNeTs, middleware architecture, 12-14
SensorWar
basic architecture, 12-11
runtime environment, 12-12
scripting language, 12-11
sensor data, fusion of, 12-10
software, aspects, 12-5
software development solutions, 12-2
software testing approach, 11-15
data collection, 11-16
emulating environment, 11-17
technology primer
ISA100, 27-22–27-24
WirelessHART, 27-20–27-22
ZigBee, 27-19–27-20
testbeds (*see* WSN testbeds)
testing methodologies
challenges associated with, 11-4
communication intricacies, 11-3
device constraints, 11-2–11-3
distributed system state, 11-3
tight integration, 11-4
test platforms
characteristics of, 11-6–11-8
DSN, 11-9–11-10
wired back-channel, 11-9
TinyOS SIMulator, 12-23
validation using testing techniques
simulators, 11-8
software testing methodologies, 11-8–11-9
test platforms, 11-5–11-8
vs. traditional distributed systems, 7-2
Wireless vibration monitoring application
communication standard and technology, 27-15–27-16
goals and requirements, 27-13–27-15
schematic representation, 27-15
sensor prototype design, 27-33
Wireline-based networks, 1-2
WiseMAC protocol, 8-9–8-10
Witness-based approach, 10-26–10-27

WSNs, *see* Wireless sensor networks
WSN testbeds
  DSN, 11-9–11-10
  integrated testing architecture
    components, 11-11
    continuous integration, 11-11–11-15
    environment influence, 11-10–11-11
    physical characterization, 11-20–11-21
    physical parameter extraction, 11-17–11-19
    physical stimulation, 11-19–11-20
    software testing approach, 11-15–11-17

X

X-by-Wire applications, TT communication systems, 13-8
X-by-Wire systems, 13-21
  dependability, 1-4
X-by-Wire technology, 13-2
XCP packet, structure, 18-13
XCP protocol stack, 18-12

Z

ZigBee, 28-5, 28-13
  architecture, 27-19
  functions, 27-19–27-20
  network devices, 27-20
  protocol stack, 27-19
  security, 27-20
  WSN standard, 27-17
ZigBee device object (ZDO), 29-27
Z-MAC protocol, 8-12–8-13

Considered a standard industry resource, the *Embedded Systems Handbook* provided researchers and technicians with the authoritative information needed to launch a wealth of diverse applications, including those in automotive electronics, industrial automated systems, and building automation and control. Now a new resource is required to report on current developments and provide a technical reference for those looking to move the field forward yet again. Divided into two volumes to accommodate this growth, the ***Embedded Systems Handbook, Second Edition*** presents a comprehensive view on this area of computer engineering with a currently appropriate emphasis on developments in networking and applications. Those experts directly involved in the creation and evolution of the ideas and technologies presented offer tutorials, research surveys, and technology overviews that explore cutting-edge developments and deployments and identify potential trends.

This second self-contained volume of the handbook, ***Networked Embedded Systems***, focuses on select application areas. It covers automotive field, industrial automation, building automation, and wireless sensor networks. This volume highlights implementations in fast-evolving areas which have not received proper coverage in other publications. Reflecting the unique functional requirements of different application areas, the contributors discuss inter-node communication aspects in the context of specific applications of networked embedded systems.

Those looking for guidance on preliminary design of embedded systems should consult the first volume: Embedded Systems Design and Verification.

K10386

ISBN: 978-1-4398-0761-3

90000

9 781439 807613

CRC Press
Taylor & Francis Group
an informa business
www.crcpress.com

6000 Broken Sound Parkway, NW
Suite 300, Boca Raton, FL 33487
270 Madison Avenue
New York, NY 10016
2 Park Square, Milton Park
Abingdon, Oxon OX14 4RN, UK